Why Workflows ?

1. Customizable business process.
   1. Rehost workflow designer in custom application.
   2. Business user can design and customize business process draging & droping specific activities on a flow chart.
2. Coordination of long-running work.
3. Transparancy.

Workflow Foundation offre un outil de design de workflow sous forme d’un *Control* utilisable sous WPF. Ce control peut être intégré dans une application permettant ainsi aux utilisateurs finaux de créer leur propres workflows à l’aide de diagrammes.

Toutefois, lorsqu’on veut créer des workflow destinés à être exécutés sur une longue période ou avec des phases d’interruption entre les différentes activités, il faut alors envisager la persistance du workflow. WF prévoit cette fonctionnalité. Ainsi on peut enregistre l’état d’un workflow en cours d’exécution et le réactiver plus tard. Le mécanisme de persistance assure la sauvegarde de tous les composant du workflow : les acticités leur statut, arguments et variables, les bookmarks, les variableS globales, les paramètres globaux, …

Il suffit d’insérer à un ou plusieurs endroits du workflow une *Activity* de type « Persist ». Mais, ça peut vite devenir fastidieux si on veut sauvegarder le workflow à chaque étape. De plus, ça nuit à la lisibilité du diagramme.

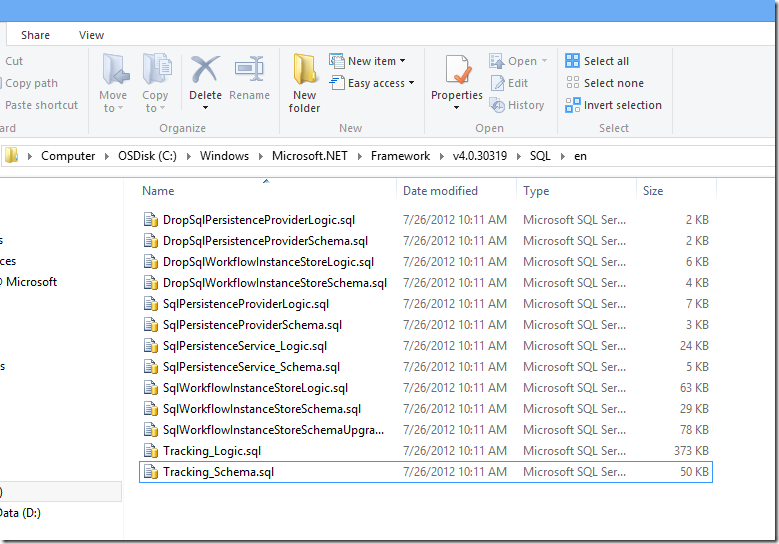
La solution est d’utiliser une *Persist Activity* dans chaque *Custom Activity*. Il suffit d’étendre une *NativeActivity* qui par ailleurs peut lancer des sous-*Activity*. Ensuite, on ajoute la *Persist Activity* dans les metadata en tant que *ImplementationChild*. La dernière étape consiste à lancer la *Persist Activity* à partir de la méthode *Execute* (peu importe à quel endroit dans la méthode car la persistance ne prendra réellement cours qu’après que l’activity soit « *Completed* »)

Il faut noter que la sortie de la méthode *Execute* ne signifie pas que l’activité soit *Completed*. Les bookmarks qui sont largement dans les *CustomActivity* , bloquent l’activité. Lorsqu’on sauvegarde un workflow , les bookmarks actifs font partie de cette sauvegarde. Cela demande la mise en place d’un mécanisme de reprise lors de la réactivation d’un workflow sauvegardé. Cet aspect va être développé plus en détail ci-dessous.

Workflow Foundation Sql Scripts

<https://blogs.msdn.microsoft.com/carlos/2013/01/10/workflow-foundation-sql-scripts/>

If you install the .Net Framework 4.5 and you look at the folder %systemroot%\Microsoft.NET\Framework\v4.0.30319\SQL\en, or %systemroot%\Microsoft.NET\Framework64\v4.0.30319\SQL\en, you’ll find a number of Sql Script files:



These scripts are all related to Workflow Foundation, and in this post I try to clarify what each script is used for.

Scripts usually come in pair, with a XxxSchema.sql file matched by a XxxLogic.sql file. The XxxSchema.sql file creates tables and views, the XxxLogic.sql file creates stored procedures, functions, triggers etc. The pairs are always meant to be executed in this order: XxxSchema.sql first, then XxxLogic.sql. Scripts in a pair should be run in the same database.

**SqlPersistenceService\_Schema.sql** and**SqlPersistenceService\_Logic.sql**

These scripts are used for WF3 persistence. You may wonder why these scripts, that refer to WF3, are in the folder of the .Net Framework 4.x. This is because WF3 is also in .Net 4.x, as I pointed out in my [previous post](http://blogs.msdn.com/b/carlos/archive/2009/08/27/workflow-foundation-migration-from-wf3-to-wf4-notes-and-references.aspx). If you are using WF3 in .Net 4, use these scripts to create the workflow persistence DB.

The DB schema for WF3 is relatively simple, with only 2 tables: dbo.InstanceState and dbo.CompletedScope. dbo.InstanceState hosts the actual persisted workflow instances, dbo.CompletedScope is used only in advanced scenarios (compensation).

Tracking\_Schema.sql and Tracking\_Logic.sql

WF3 ships with a tracking service, System.Workflow.Runtime.Tracking.SqlTrackingService, that writes information to a Sql Server database. You create such a database by running Tracking\_Schema.sql and Tracking\_Logic.sql. You need the tracking database only if you are using SqlTrackingService. It is possible, though not mandatory, to use the same database for persistence and tracking. In other words, you can run SqlPersistenceService\_Schema.sql SqlPersistenceService\_Logic.sql, Tracking\_Schema.sql and Tracking\_Logic.sql in the same database.

WF4 has a different tracking architecture. Also, it ships a tracking participant, EtwTrackingParticipant, that tracks information to .etl files, not to Sql Server. Consequently, there is no equivalent of Tracking\_Schema.sql and Tracking\_Logic.sql in WF4.

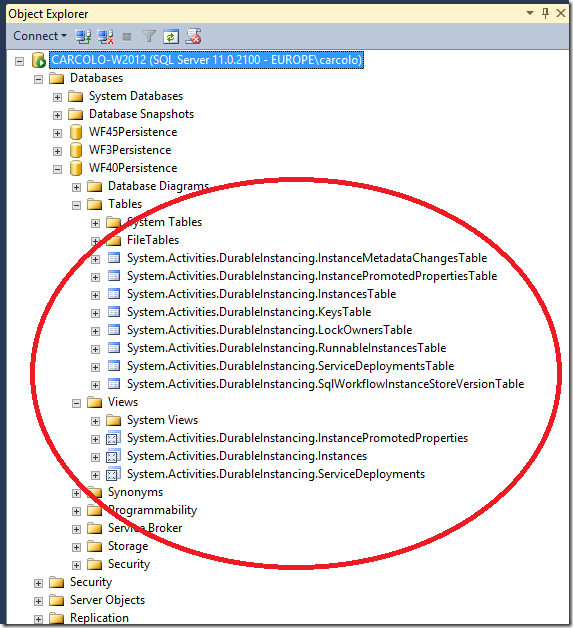
SqlPersistenceProviderSchema.sql and SqlPersistenceProviderLogic.sql

WF in the .Net Framework 3.5 (WF3.5) adds support for [Workflow Durable Services](http://msdn.microsoft.com/en-us/library/vstudio/bb412197(v=vs.90).aspx), and this requires an update to both the schema and the logic of the database. SqlPersistenceProviderSchema.sql and SqlPersistenceProviderLogic.sql update the persistence database accordingly. Note that these scripts should be executed in an existing WF3 persistence database, that is a database where SqlPersistenceService\_Schema.sql and SqlPersistenceService\_Logic.sql have already been run.

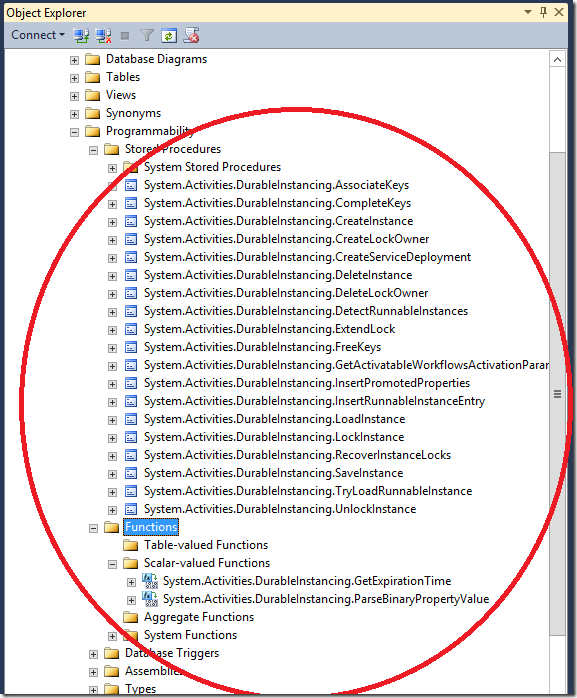
SqlPersistenceProviderSchema.sql adds the dbo.InstanceData table. SqlPersistenceProviderLogic.sql adds the dbo.DeleteInstance, dbo.InsertInstance, dbo.LoadInstance, dbo.UnlockInstance and dbo.UpdateInstance stored procedures.

SqlWorkflowInstanceStoreSchema.sql and SqlWorkflowInstanceStoreLogic.sql

WF4 introduces a completely new object model, and with it comes a different schema of the persistence database. SqlWorkflowInstanceStoreSchema.sql and SqlWorkflowInstanceStoreLogic.sql create a database suitable for persisting WF4 workflows. The script files are named after the class in WF4 that implements workflow persistence in SqlServer:,System.Activities.DurableInstancing.SqlWorkflowInstanceStore, The structure of the WF4 persistence database is more complex than the structure of the WF3 database. SqlWorkflowInstanceStoreSchema.sql creates a number of tables and views:



SqlWorkflowInstanceStoreLogic.sql adds stored procedures and functions:



SqlWorkflowInstanceStoreSchemaUpgrade.sql

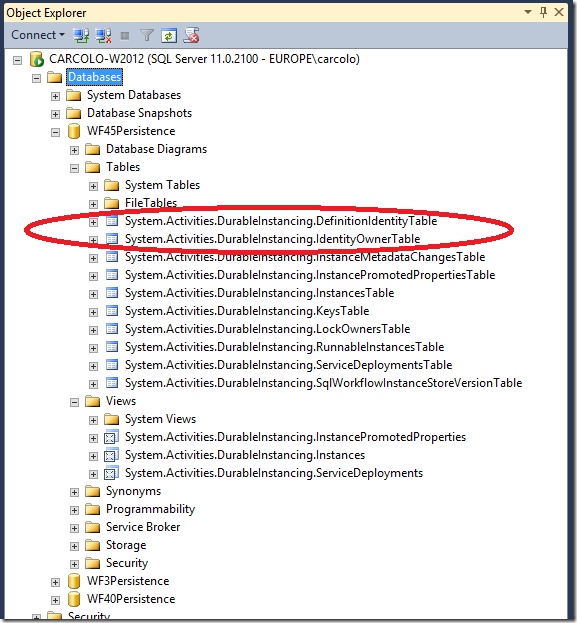
WF4.5 introduces [versioning](http://msdn.microsoft.com/en-us/library/hh314054.aspx). SqlWorkflowInstanceStoreSchemaUpgrade.sql updates the database created by SqlWorkflowInstanceStoreSchema.sql and SqlWorkflowInstanceStoreLogic.sql to accommodate for versioning information. The step-by-step procedure is documented [here](http://msdn.microsoft.com/en-us/library/hh314054.aspx#UpdatingWF4PersistenceDatabases).

Note that you can save and reload WF4.5 workflows in WF4 databases (that is, databases created with SqlWorkflowInstanceStoreSchema.sql and SqlWorkflowInstanceStoreLogic.sql, but without SqlWorkflowInstanceStoreSchemaUpgrade.sql), as long as you do not associate versioning information to them.

The upgrade of the database consists of these main steps:

* New tables are created to handle versioning: DefinitionIdentityTable and IdentityOwnerTable
* Existing tables are altered with additional columns: InstancesTable.SurrogateIdentityId, RunnableInstancesTable.SurrogateIdentityId, LockOwnersTable.WorkflowIdentityFilter,
* Values are inserted in new tables and in new columns of existing tables, to represent standard versioning information for the workflow instances persisted in the database
* New views and stored procedures are created.

After executing the script, you’ll see the new tables:



The set of stored procedures and functions is the same of the WF4 version.

DropXxx.sql

The DropXxx.sql scripts drop tables and other artifacts. They are useful when a database hosts something else, in addition to persistence or tracking information. In that case, deleting the entire database would cause loss of data.

The versions of SqlWorkflowInstanceStoreXxx.sql

There is one further consideration that applies to SqlWorkflowInstanceStoreSchema.sql and SqlWorkflowInstanceStoreLogic.sql: if you compare these scripts on a machine with the .Net Framework 4.0, and on a machine with the .Net Framework 4.5 installed, you’ll find that they are different. This is because *the .Net Framework 4.5 ships with versions of these files that directly create a WF4.5 database*. For instance, SqlWorkflowInstanceStoreSchema.sql in the 4.5 version of the .Net Framework creates the  DefinitionIdentityTable, whereas the SqlWorkflowInstanceStoreSchema.sql in the 4.0 version of the .Net Framework does not.

Why is that? Well, if you installed the 4.5 version of the framework and subsequently create a WF Persistence Database, it makes sense to create it with a structure that is suitable for WF4.5 workflows. Note that SqlWorkflowInstanceStoreSchemaUpgrade.sql still has its own place: if you have an existing WF4 persistence database, you should run SqlWorkflowInstanceStoreSchemaUpgrade.sql against it, instead of running the 4.5 version of SqlWorkflowInstanceStoreSchema.sql and SqlWorkflowInstanceStoreLogic.sql: this way, you would preserve the workflow instances currently persisted in the database.

Which scripts to use?

Here is a quick summary of which scripts you need, depending on the version of WF that you are using:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **WF3** | **WF3.5** | **WF4** | **WF4.5** |
| **SqlPersistenceService\_Schema.sql  SqlPersistenceService\_Logic.sql** | X | X |  |  |
| **SqlPersistenceProviderSchema.sql  SqlPersistenceProviderLogic.sql** |  | X |  |  |
| **Tracking\_Schema.sql  Tracking\_Logic.sql** | X | X |  |  |
| **SqlWorkflowInstanceStoreSchema.sql  SqlWorkflowInstanceStoreLogic.sql** |  |  | X | X |
| **SqlWorkflowInstanceStoreSchemaUpgrade.sql** |  |  |  | X |

Scripts that are used by WF3 are also found in previous versions of the framework. So it should not come as a surprise to you that, under the folder %systemroot%\Microsoft.NET\Framework\v3.0\Windows Workflow Foundation\SQL\en, you find 4 script files: SqlPersistenceService\_Schema.sql, SqlPersistenceService\_Logic.sql, Tracking\_Schema.sql and Tracking\_Logic.sql. Likewise, you’ll find SqlPersistenceProviderSchema.sql and SqlPersistenceProviderLogic.sql in the %systemroot%\Microsoft.NET\Framework\v3.5\SQL\EN folder. And. since WF3 that ships with the .Net Framework 4.x is the same that ships with .Net Framework 3.x, the script files with the same name on different folders are identical, save for references to assemblies: Tracking\_Schema.sql in %systemroot%\Microsoft.NET\Framework\v3.0\Windows Workflow Foundation\SQL\en references mscorlib version 2.0.0.0 and System.Workflow.ComponentModel version 3.0.0.0, whereas Tracking\_Schema.sql in %systemroot%\Microsoft.NET\Framework64\v4.0.30319\SQL\en references mscorlib version 4.0.0.0 and System.Workflow.ComponentModel version 4.0.0.0.