

ECO quick start 06 - Making the EcoSpace persistent

Table of Contents

| | |
|--------------------------------|----------|
| Overview | 1 |
| Prerequisites and goals | 2 |
| XML persistence | 3 |
| SQL Server persistence | 4 |
| Inspecting the SQL database | 5 |
| Polymorphism | 7 |
| Summary | 9 |
| Index | a |

1 Overview

Until persistence is added all changes to objects in the EcoSpace will be lost when the application terminates. Prolonging the life of these objects beyond the life of the application is a very simple process. By adding an ECO persistence component to the EcoSpace it is possible to update a persistence storage with changes made during the application's execution, this persistence storage might be a local XML file, a number of supported database servers, or a custom persistence mechanism to work with persistence storages that are not supported by default.

2 Prerequisites and goals

Prerequisites

To successfully follow this document the user should have read the preceding articles in this series and have a saved project.

Goals

By the end of this document you be able to

- Add persistence to an EcoSpace.
- Use an XML persistence store.
- Use an RDBMS persistence store.

3 XML persistence

The PersistenceMapperXml component is the same component used by the Quick Prototype tool to persist data between testing phases. This persistence mapper will save objects in the EcoSpace to a specific XML formatted file each time the application instructs the EcoSpace to do so. As the entire XML file is read when the EcoSpace is activated and rewritten during saving and this persistence component is really only suitable for use during the development phase, a power failure during the process of updating the file would result in data loss.

1. Open the QuickStartVCL project.
2. In the Project Manager window double-click the node entitled "PersistenceMapperProvider".
3. When the component's designer appears, drop a PersistenceMapperXml component onto it.
4. Set the mapper's "FileName" property to "TestData.xml"
5. Click the PersistenceMapperProvider's design surface to select it in the Object Inspector.
6. Ensure its "PersistenceMapper" property is set to "PersistenceMapperXml1".
7. Run the application.
8. Add a new Author.
9. Click the "Post" button on the navigator.
10. Click the "Commit" button to save all changes to disk.
11. Restart the application.
12. Note how the objects reappear from the last time the application was executed.

4 SQL Server persistence

The PersistenceMapperSqlServer component allows changes to be persisted to a SQL Server database server. It is possible for ECO to create and maintain the database structure, or to provide object to table mapping information for ECO to update a manually maintained database structure. In this article ECO will be used to create a database with a default structure. This is by far the quickest way of developing applications as it removes the necessity for the developer to maintain the database structure and mapping information.

1. Delete the PersistenceMapperXml1 component from the PersistenceMapperProvider.
2. Create a new SQL Server database named "EcoQuickStart".
3. Drop a SqlConnection component onto the PersistenceMapperProvider.
4. Set the connection string of the SqlConnection component to point to the "EcoQuickStart" database.

NOTE: During design time it is recommended that you use the system administrator account "sa", but at runtime you should set your connection string to use a non administrator account that has been granted access to the database.

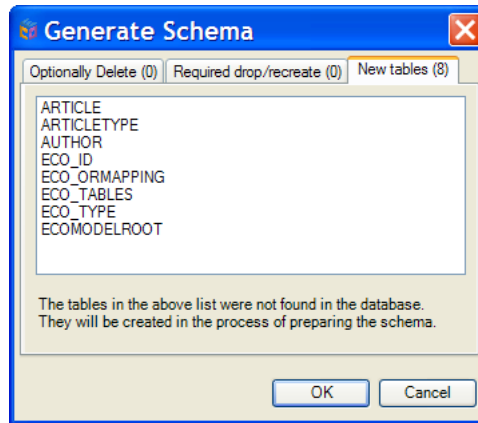
5. Drop a PersistenceMapperSqlServer component onto the PersistenceMapperProvider's design surface.
6. Ensure its "Connection" property is set to "SqlConnection1".
7. Right-click PersistenceMapperSqlServer1 and from the context menu select "SQL Server 2000/2005 setup", this will configure the component so that it is aware of the target database server's reserved words etc.
8. Select the PersistenceMapperProvider in the Object Inspector and ensure that its "PersistenceMapper" property is set to "PersistenceMapperSqlServer1".

The EcoSpace is now configured to use the SQL Server database for persistence. As this is an empty database some tables will be needed for storing the objects.

7. Compile your project to ensure the generated binaries reflect any changes to your model.
8. At the bottom of the PersistenceMapperProvider there is a selection of icons that perform different actions. Click the "Generate Schema" icon.



9. A form will now appear listing tables to delete, tables to recreate, and new tables to be created. Selecting the [New tables] tab will reveal the following tables.

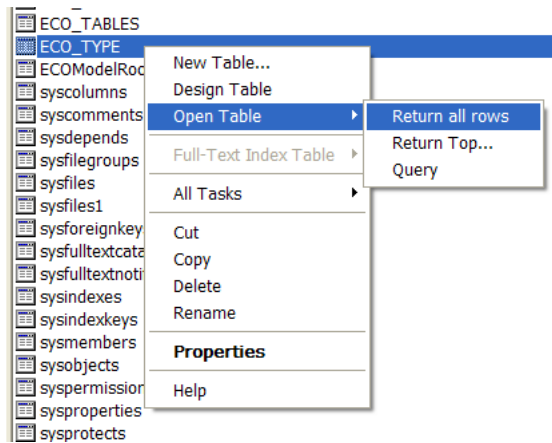


| Table name | Description |
|---------------|--|
| Article | This table will hold instances of the Article class. |
| ArticleType | This table will hold instances of the ArticleType class. |
| Author | This table will hold instances of the Author class. |
| ECO_ID | This table will contain a single row holding the next Integer value to use as the primary key for new objects. |
| ECO_ORMAPPING | This table will contain a single row holding XML information describing the object to database mapping information. |
| ECO_TABLES | This table will contain a list of tables created by ECO. |
| ECO_TYPE | This table will contain a list of modeled class names mapped to integer values, these are used to determine the exact type of an object in a row when inheritance is used. |
| ECOModelRoot | This table will hold instances of the ECOModelRoot class, this class is ultimately the superclass of all classes within the model. This table holds the ID of the object and an integer value identifying the class type (see ECO_TYPE). |

10. Click "OK", the database structure will now be updated.
11. Run the application.
12. Add a new Author.
13. Click the "Post" button on the navigator.
14. Click the "Commit" button to save all changes to disk.
15. Restart the application.
16. Note how the objects reappear from the last time the application was executed.

4.1 Inspecting the SQL database

1. Using your SQL Server management tool view the tables within the EcoQuickStart database.
2. Right-click the table ECO_TYPE, from the context menu select "Open Table" followed by "Return all rows".



3. You will now be presented with the following data:

| ECO_TYPE | CLASSNAME |
|----------|--------------|
| 0 | Author |
| 1 | Article |
| 2 | ArticleType |
| 3 | ECOModelRoot |

From this data you can see that the Author class has an ECO_TYPE id of "0" and ArticleType has an ECO_TYPE id of "2".

4. Next view all data rows for the ECOModelRoot table:

| ECO_ID | ECO_TYPE |
|--------|----------|
| 1 | 2 |
| 2 | 2 |
| 3 | 2 |
| 4 | 0 |

5. From this information we can determine that there are 4 object instances in the entire database. By inspecting the ECO_TYPE column we can see that there are three ArticleType instances and one Author instance.

6. View all data rows for the ArticleType table to confirm this assumption.

| ECO_ID | ECO_TYPE | Name | ID |
|--------|----------|----------|----|
| 1 | 2 | VCL.NET | 1 |
| 2 | 2 | ASP.NET | 2 |
| 3 | 2 | WinForms | 3 |

7. View all data rows for the Author table to see data entered earlier.

| ECO_ID | ECO_TYPE | AutoPublishArticles | FirstName | Salutation | Password | ID | EmailAddress | LastName |
|--------|----------|---------------------|-----------|------------|----------|----|--------------|----------|
| 0 | 0 | 0 | Peter | Mr | password | 1 | me@home.com | Morris |

4.2 Polymorphism

The Latin word "Polymorph" means to have many forms. As ECO allows you to model classes that descend from other classes it is obviously possible via object oriented programming techniques to use polymorphic techniques. For example

- There is a class named "Task".
- There is a descendant of Task named "BackupDatabaseTask".
- There is another descendant of Task named "RestoreDatabaseTask".

It would be possible to pass an instance of either a BackupDatabaseTask or RestoreDatabaseTask to any method expecting an instance of Task.

ECO allows the use of such polymorphic techniques when modeling associations between classes, and also when retrieving object instances from the persistent storage.

- An association between, for example, a Schedule class and the Task class would allow the application to not only add instances of Task to Schedule.Tasks but also instances of any class descended from Task. This would allow Schedule.Tasks to contain both BackupDatabaseTask and RestoreDatabaseTask.
- When retrieving object instances from the persistent storage the correct type of object is instantiated. Asking for "Task.allInstances" would return a collection of object instances which contains both BackupDatabaseTask and RestoreDatabaseTask objects.

When generating a table in the database for a class with no ancestor ECO will create an additional column named ECO_TYPE;

| ECO_ID | ECO_TYPE | Name | ID |
|--------|----------|----------|----|
| 2 | 2 | VCL.NET | 1 |
| 2 | 2 | ASP.NET | 2 |
| 3 | 2 | WinForms | 3 |

this identifies the type of the object according to the ECO_TYPE table.

| ECO_TYPE | CLASSNAME |
|----------|--------------|
| 0 | Author |
| 1 | Article |
| 2 | ArticleType |
| 3 | ECOModelRoot |

When generating a table in the database for a class with an ancestor this `ECO_TYPE` column will not be generated. Instead the object instance's data will be split across two tables, for example `Task` and `BackupDatabaseTask`. This approach enables the developer to model the application's business classes as if they were part of some kind of persistent memory rather than persisted to a database.

NOTE: It is possible to control the database schema in a number of ways:

- Specifying custom key type, a GUID for example.
- Specifying custom mapping information on a class to identify whether it should have its own table, add its required columns to its parent class's table, or add its required columns to its descendant class's tables (abstract classes only)

5 Summary

Making an EcoSpace persistent is a very simple process. As a consequence switching to a new database can be achieved in seconds. It is also possible with a small amount of programming to identify which type of database server to connect to at runtime, allowing your application to connect to the database of your customer's choice.

Database structure management

ECO is not only capable of creating your database structure, but also updating the database structure to reflect changes made to your business model. Automatic database structure management saves the developer a considerable amount of time when creating applications.

Supported databases

At the time of writing the following databases are supported by ECO.

- BlackFish.
- DB2 (via Delphi BDP or DBX components).
- InterBase (via Delphi BDP or DBX components).
- FireBird 2.
- MySQL.
- Oracle.
- SQL Server.
- Sybase (via Delphi BDP or DBX components).
- XML file.

More advanced persistence techniques such as mapping to an existing database, reverse engineering an existing database, and using a remote persistence server will be covered in a subsequent document.

Index

I

Inspecting the SQL database 5

O

Overview 1

P

Polymorphism 7

Prerequisites and goals 2

S

SQL Server persistence 4

Summary 9

X

XML persistence 3