Sheridan | Get Creative

Advanced .NET Server Development: Object-Relational Mapping (ORM)

Instructor: Amandeep S. Patti

Learning Outcome

- Explain the purpose of Object Relational Mappers

Objectives

- Introduction to ORMs
- Introduction to .NET ORMs
- Benefits of ORMs
- Selecting a suitable .NET ORM

Introduction to ORMs

- Object-relational mapping is a programming technique
  - for converting data between incompatible type systems
  - by creating objects using OO programming languages
  - essentially wrapping tables or stored procedures in classes and interact with database using their methods and properties

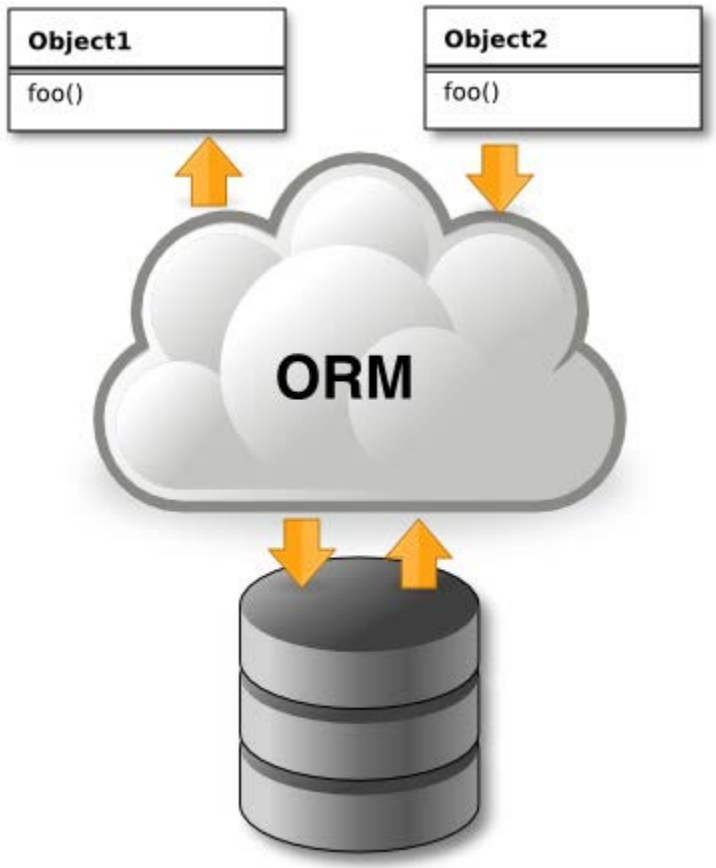
- ORM is an automated way of
  - connecting an object model, sometimes referred to as a domain model, to a relational database by using metadata as the descriptor of the object and data.

- This creates, in effect, a "virtual object database"

Introduction to ORMs

- ORM is the act of
  - connecting object code, whether it is in C#, Java, or any other object-oriented language, to a relational database.
- This act of mapping is an efficient way to overcome the mismatch that exists between object-oriented development languages and relational databases.
  - Such a mismatch can be classified as an inequality between the native object-oriented language operations and functions and those of a relational database.

Introduction to ORMs



Relational Models vs. OO models:

- Types (ex. char(n) versus string)
- Identity (keys versus address or equality)
- Relations (foreign keys versus references)
- Many to many relations require linking table in SQL
- Inheritance and polymorphism

Benefits of ORMs

- work in the OO model
  - without having to worry about the underlying data structure
- Allow us to easily save objects to the database and load them from the database
- Typically provide support for the full set of CRUD (create, read, update, delete) operations
- Save time and money (time to market)
- Focus on the business logic
  - rather than database/persistence logic

Introduction to .NET ORMs

- Nhibernate
  - First major ORM for .NET
  - An open source solution which is still around
  - http://nhibernate.info
- LINQ to SQL
  - First ORM provided by .NET framework
  - http://msdn.microsoft.com/en-ca/library/bb425822.aspx
- Entity Framework
  - Replaced LINQ to SQL
  - http://msdn.microsoft.com/en-ca/data/ef.aspx
- Other ORMs for .NET
  - SubSonic, DataObjects.Net, Telerik's OpenAccess ORM, BLToolkit, LBLGen Pro, LINQConnect, MyBatis.NET, etc.

Selecting a suitable .NET ORM

- Free or Open Source or Commercial
- Support given by the company or community
- Handle enterprise class ORM problems
- Support LINQ
- Out of the box (default) database support
- Support querying of views, stored procedure or functions
- Caching mechanism
- Support lazy-loading
- Support batching mode
  - updating many items at once

Selecting a suitable .NET ORM

- The ORM tool you choose to use should be evaluated based on a set of criteria that meets your goals; however, the following are good starting points:
  - object-to-data-base mapping
  - object caching
  - GUI mapping
  - portability (multiple DB support)
  - dynamic querying
  - lazy loading
  - nonintrusive persistence
  - code generation
  - stored procedure support

Videos

- Choosing tools for managing data: Object-relational mapping (ORM) tools and the Entity Framework
  - http://www.lynda.com/ASPNET-tutorials/Choosing-tools-managing-data-Object-relational-mapping-ORM-tools-Entity-Framework/158377/171748-4.html

Citation

- Mehta, Vijay P.. "Chapter 1 - Getting Started with Object-Relational Mapping". *Pro LINQ Object Relational Mapping with C# 2008*. Apress. © 2008.