



Entity Framework 4.0

Stefano Paluello
 TTG – Torino Technologies Group
 stefano.paluello@pastesoft.com
 (@palutz)



Microsoft | TechNet

Agenda

- Data Access in the history
- EF4 Overview
- EF Pattern and Developing
- How to query EF4



Microsoft | TechNet

It was a long path...

DATA ACCESS IN THE HISTORY



Microsoft | TechNet

Accessing before '90s

- RAW Data, Direct APIs

Accessing data in 1990

- ODBC (abstract call-level)

Accessing pre .Net

- OLE DB, ADO (object level)

Accessing data in .Net

- ADO.NET, Datasets, DataReaders

Today

- Ling, Entity Framework... O/RM?



Microsoft | TechNet

What is an O/RM?

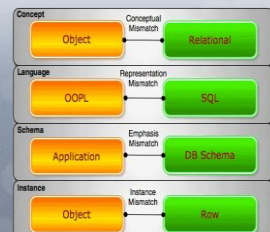
- **Object**
- **Relational**
- **Mapping**
- An *abstraction*, a technique for converting data between incompatible type systems, the RDBMS and OOP (aka impedance mismatch), hiding the complexity of the underlying tables



Microsoft | TechNet

Impedance.. what??

- The object-relational **impedance mismatch** is a set of conceptual and technical difficulties that are often encountered when a RDBMS is being used by a program written in an OOP language or style; (Wikipedia)



Microsoft | TechNet

Why use an O/RM?

- Increase the Developer productivity
 - Better, faster and less code than “average” developer
- Impedance mismatch
- Database Portability



Microsoft | TechNet

O/RM is not...

- The solution for all our problems
- The fastest way to do anything
- A cool new stuff that we must have in our application (it won't help us to hang out with our new sexy colleague or desk-mate ☺)



Microsoft | TechNet

Entity Framework 4.0

OVERVIEW



Microsoft | TechNet

The “managed” path to EF

- Typed Dataset – shipped ☺
- ObjectSpaces v1, v2 – never shipped ☹
- MS Business Framework – never shipped ☹
- WinFS – never shipped ☹
- LINQ to SQL – shipped ☺ (.Net 3.5)
- LINQ to Entities and Entity Framework v.1.0 – shipped ☺ (.Net 3.5 SP1)
- Entity Framework 4.0 – shipped ☺ (.Net 4.0)



Microsoft | TechNet

The new kid on the block: ADO.net Entity Framework

- EF is the O/RM in the .NET Framework
- Microsoft's strategic technology
- Used in other Microsoft products:
 - WCF Data services, Azure Table Storage, Sharepoint 2010, SQL Server Reporting Services and PowerPivot for Excel, ...
- Big investment in this release (.Net 4.0)
 - took the best from LINQ to SQL
- Database and ORM (!!!) vendors supporting it (IBM, OpenLink, DevForce, LLBLGen ...)



Microsoft | TechNet

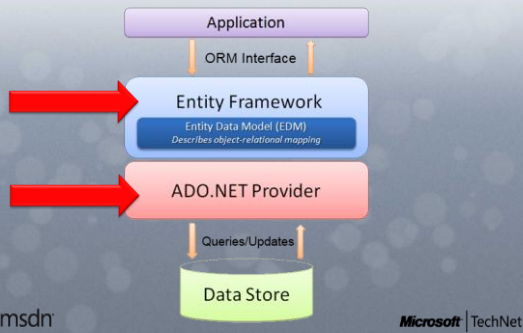
What about my “old” ADO.Net code?

- ADO.NET Core provides the most control
- DataSets aren't going away
- ADO.NET Core is the foundation of the Entity Framework
 - Provider layer
 - Incremental evolution for apps from ADO.NET Core to EF

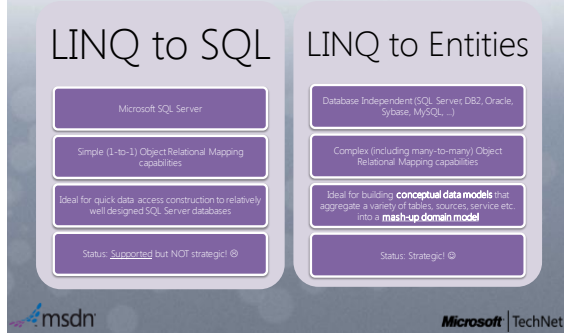


Microsoft | TechNet

A typical scenario with EF



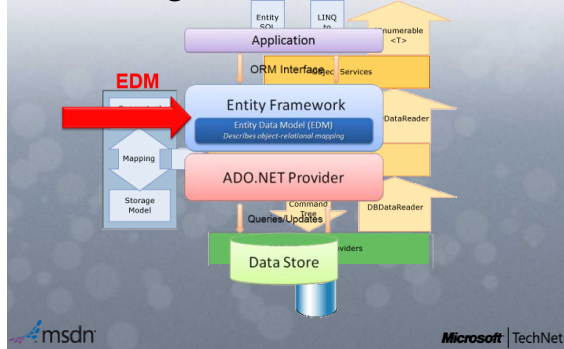
Linq To SQL vs Linq To Entities



What is Entity Framework?

- Data access framework
- Supports data-centric applications and services
- Enables programming against a conceptual application model
- Enables independency of any data storage engine or relational schema

Let's dig into Entity Framework



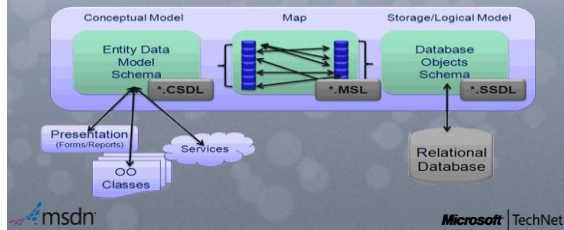
EDM – Entity Data Model

- Invented by Dr. Peter Chen (1970s) and named ERM (Entity Relationship Model)
- ERM
 - Conceptual Layer
 - Mapping Layer
 - Storage Layer
- Now: EDM in Entity Framework



Entity Data Model

- Comprised of three layers (CSDL, MSL, SSDL)
- Database agnostic
- Comprised of Entities, Associations, Functions



Entity Data Model in .Net

- It's actually an XML file (.edmx)

```
<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="2.0" xmlns:edmx="http://schemas.microsoft.com/ado/2008/10/edmx">
  <!-- EF Runtime content -->
  <edmx:Runtime>
    <!-- SSDL content -->
    <edmx:StorageModels>...</edmx:StorageModels>
    <!-- CSDL content -->
    <edmx:ConceptualModels>...</edmx:ConceptualModels>
    <!-- C-S mapping content -->
    <edmx:Mappings>...</edmx:Mappings>
  </edmx:Runtime>
  <!-- EF Designer content (DO NOT EDIT MANUALLY BELOW HERE) -->
  <Designer xmlns="http://schemas...">...</Designer>
</edmx:Edmx>
```

msdn

Microsoft | TechNet

A glimpse to Entity Framework 4

DEMO

msdn

Microsoft | TechNet

What's new in EF4

- Pluralization
- Foreign Keys in the Conceptual Model
- Stored Procedures
- Self tracking Entities and new methods for N-Tier App. Dev.
- Model First
- Complex Types
- Model Defined Functions
- Code Generation
- Persistence Ignorance
- Lazy Loading
- Code Only
- EntityDataSource support for QueryExtender control
- Testability with IOObjectSet<T>
- Direct execution of Store Commands fromObjectContext
- Functions in LINQ to Entities queries
- OrderBy improvement in LINQ to Entities
- Customized Object-Layer code generation
- Entity Designer extensibility
- Entity Data Model Wizard improvement (naming service)
- ...
- Entity Framework Futures

msdn

Microsoft | TechNet

Code Generation (T4)

- Entity Framework 4 shipped with a number of T4 code-generation templates which you can customize or replace with your own. (T4 is a code-generation technology built into Visual Studio 2008 or later)
- The whole point of using T4 for code-generation is that it makes easy to customize the generated entities, like when we need to add some particular validation on setting up entity properties.

msdn

Microsoft | TechNet

Entity Designer Improvements

Complex Type support

Create complex types in your Model

Foreign Keys

Optionally include foreign keys in your model

Pluralization

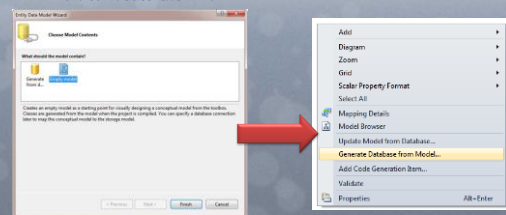
Better naming management of the Entity Set

msdn

Microsoft | TechNet

Entity Designer Improvements

- **Model First:** first design then create the DB!
Start with an Empty Model and create you concept model (or Domain Model) adding Entities and Associations. Then ask EF4 to create the DB.



msdn

Microsoft | TechNet

Model First Demo

DEMO

Microsoft | TechNet

EF4 creating models Recap

- Start with the **database** ...
 - ... and then generate the model + code
 - Scenario: DB already exists, or you want low level control over it
 - How to: Import model into edmx and tweak
- Start with an **edmx model** ...
 - ... and then generate the database + code
 - Scenario: You want separation from code and database in a declarative format
 - How to: Create a model in the designer
- Start with **.net classes** ...
 - ... and then generate the database + model
 - Scenario: Primarily focused on code shape, database is an implementation detail
 - How to: Define classes in code and then adjust shape using contextbuilder



Microsoft | TechNet

EF Model Mapping capabilities

- **Inheritance**
 - Table per Hierarchy
 - Table per Type
 - Table per Concrete Type
 - Hybrids
- **Many entities to one table**
- **Stored Procedures**
- Many tables to one entity
- **Abstract Entities**
- Associations within EntitySets
- Associations across EntitySets
- Store-side discriminators
- EDM-side discriminators
- QueryViews, Defining Query, CommandText



Microsoft | TechNet

EF4 Patterns

EF4 DEVELOPING

Microsoft | TechNet

EF4 Patterns

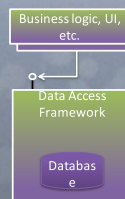
- Repository, Unit of Work, Identity Map
- POCO (Plain Old CLR Object) and Proxies
- Lazy/Deferred Loading
- Foreign Keys
- Object Set and Testability



Microsoft | TechNet

Repository Pattern

- *"Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects"*
- **Why do I need it?**
 - Separation of concerns
 - Greater control
 - Easier testability



- **How to:**
 - Simple class & interface
 - Just a collection



Microsoft | TechNet

Repository Pattern with EF4

DEMO

msdn

Microsoft | TechNet

Repository Pattern Recap

- Repositories are easy with EF!
- Best Practices:
 - Make the repository's interface as simple as possible
 - Create separate repositories for each **"aggregate"**
 - Keep other layers **"persistence ignorant"**
 - Multiple repositories can share sameObjectContext as **"unit of work"**
- Popular additions:
 - Query methods return IQueryable<T> (be careful, you could query your entities on the UI with this)
 - Using **"specification pattern"**, e.g. repository.Find(c=>c.Id == id)
 - Repository<T> base class (don't repeat yourself)

msdn

Microsoft | TechNet

Unit of Work Pattern

"Maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems."

It means that a **Unit of Work** is a context/session/unit object that tracks the changes of business entities during one business transaction.

```
public interface IUnitOfWork<T>{
    void RegisterNew(T entity);
    void RegisterDirty(T entity);
    void RegisterClean(T entity);
    void RegisterDeleted(T entity);
    void Commit();
}
```

msdn

Microsoft | TechNet

POCO & Persistence Ignorance

- With EF4 you can now define your own objects that are completely decoupled from persistence concerns.
- These objects don't need to be a subclass of *EntityObject* or to implement a set of IPOCO interfaces (i.e. *IEntityWithKey*, *IEntityWithChangeTracker* and *IEntityWithRelationships*), like it was with the previous version of EF.

msdn

Microsoft | TechNet

Example of POCOs in EF4

```
public class Customer {
    public string CustomerID { get; set; }
    public string ContactName { get; set; }
    public string City { get; set; }
    public List<Order> Orders { get; set; }
}
```

```
public class Order {
    public int OrderID { get; set; }
    public Customer Customer { get; set; }
    public DateTime OrderDate { get; set; }
}
```

msdn

Microsoft | TechNet

Adding POCOs to the Context

```
public class MyContext : ObjectContext {
    private ObjectSet<Category> _categories;
    private ObjectSet<Product> _products;

    public MyContext() : base("name=MyEntities", "MyEntities") {
        _categories = CreateObjectSet<Category>();
        _products = CreateObjectSet<Product>();
    }

    public ObjectSet<Category> Categories {
        get { return _categories; }
    }

    public ObjectSet<Product> Products {
        get { return _products; }
    }
}
```

msdn

Microsoft | TechNet

Consume POCOs in my app.

```

...
MyContext db = new MyContext();

var smartphones = from p in db.Products
                  where p.Category.CategoryName == "Smartphone"
                  select p;
...

```



Microsoft | TechNet

Lazy/Deferred Loading

- Lazy Loading is aObjectContext setting, not an application setting and it works with code-generated entities as well with POCO entities.
- It's turned off by default, but is *true* on the EDM Context
- In addition to eager (*Include()*) and explicit (*Load()*) loading, related entities can be loaded automatically on demand.
- How to enable Lazy Loading:
 - *context.ContextOptions.DeferredLoadingEnabled=true*
 - Declare the property that you would like to load lazily as **virtual**. These properties can be any collection type that implements **ICollection<T>** or they can be a reference representing a 1/0..1 relationship.



Microsoft | TechNet

Object Set and Testability

- The new *ObjectSet<T>* class derives from *ObjectQuery<T>* and represents a "root" query object for an EntitySet, that also has *AddObject*, *DeleteObject* and *Attach* methods.
- In order to *improve testability*, an *IObjectSet<T>* interface was defined, that derives from *IQueryable<T>*. *IObjectSet<T>* happens to be super easy to implement as an in-memory fake object.
- Provided that your queries and CRUD operations can refer to instances of *IObjectSet<T>*, your code will now be easier and faster to test.



Microsoft | TechNet

Testability with EF4

DEMO



Microsoft | TechNet

Testability with EF4 Recap

- EF4 has a focus on testability (it's quite easy)
- Best practices:
 - Build simple repository interface
 - Swap repository with in-memory test
 - Alternate route: define simple interface for context
 - Still run tests against database often (integration test – lower but safer)



Microsoft | TechNet

Querying the Model

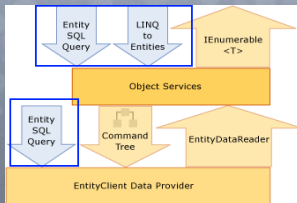
HOW TO QUERY EF4



Microsoft | TechNet

Querying the model

- Three kinds of queries in EF:
 - LINQ to Entities
 - Entity SQL with Object Services
 - Entity SQL with Entity Client



msdn

Microsoft | TechNet

LINQ to Entities

- Queries written in LINQ syntax
- Support for LINQ features
- Full IntelliSense support

```

var person = from people in context.Anagraficas
              where people.Name.StartsWith("S")
              orderby people.Age ascending
              select new
              {
                  Name = people.Name,
                  Surname = people.Surname
              };
  
```

msdn

Microsoft | TechNet

Entity SQL ObjectServices

- T-SQL-like query language
- Can query the EDM
- EF translates Entity SQL into storage-specific queries

```

var queryStr = @"SELECT NAME n
                FROM context.Anagraficas as anag
                WHERE anag.Name='Stefano'";
var person =
context.CreateQuery<Anagrafica>(queryStr);
  
```

msdn

Microsoft | TechNet

Entity Client

- ADO.Net "old"-style queries
- Prime choice for developers migrating from Ado.Net

```

using (EntityConnection conn = new EntityConnection("...") {
    conn.Open();
    using (EntityCommand cmd = conn.CreateCommand()) {
        cmd.CommandText = "SELECT * FROM Context.Anagraficas";
        using (EntityDataReader rdr = cmd.ExecuteReader(
            CommandBehavior.SequentialAccess)) {
            while (rdr.Read()) {...}
        }
    }
}
  
```

msdn

Microsoft | TechNet

Entity Client (2)

- Familiar ADO.NET object model:
 - EntityCommand
 - EntityConnection
 - EntityDataReader
 - EntityParameter
 - EntityTransaction
- Text-based results, we need EntityDataReader to read
- Read-only access to EDM. To modify data we need ObjectServices with Entity SQL

msdn

Microsoft | TechNet

Querying the model Recap

Method	Syntax	Return type	Benefits	Performance
LINQ to Entities	LINQ	Entity Objects or Anonymous Type	Object Context w/change tracking; Intellisense; Language Integration	★
EntitySQL (ObjectServices)	Entity SQL	Entity Objects or dbDataRecords	Object Context w/change tracking; Build Dynamic Queries;	★★★
EntityClient	Entity SQL	dbDataReader	Plug into existing DAL	★★★★

msdn

Microsoft | TechNet

What can I use to access data?

DATA ACCESS RECAP



Microsoft | TechNet

Native Data Access Guidance

- For new native applications wanting a generalized abstraction layer with support for multiple data sources, use **ODBC**. This is the most efficient, full-featured API in which Microsoft will continue to invest.
- For specialized, high-performance data access to SQL Server, use **SQL Server Native Client (C/C++)**, the **JDBC driver** (Java/JavaScript), or the **PHP driver** depending on your language of choice.
- If you're invested in VBA and ASP classic, continue to use **ADO**. ADO will continue to be supported (including security fixes) but won't see new feature work.
- If you really want COM-based data access, use **OLEDB**. This will also continue to be supported but won't have new feature work.



Microsoft | TechNet

.NET Data Access Guidance

- New applications should start by looking at the **ADO.NET EF4** (including LINQ to Entities) for data access.
 - Scenarios for which ADO.NET DataSets have been used are generally handled well with the Entity Framework also.
 - The EF can be incrementally adopted in applications using ADO.NET Core. For example, much of the connect/query/process code of ADO.NET Core can be easily replaced with very simple, entry-level usage of EF.
- Use **ADO.NET Core** when you want the lowest level of control.
 - ADO.NET Core remains the basis for data access in .NET.
 - Provides the most common and familiar development patterns for data access (connect, query, process)
 - DataSets and LINQ to DataSet will continue to be supported.
 - For simple applications where you don't need more than simple connections and streaming results, ADO.NET may be a good choice to get a job done quickly.
 - If you have requirements for fine-grained control, ADO.NET might give you more capabilities than the Entity Framework.
- **LINQ to SQL** is and will continue to be supported but will see little new investment.



Microsoft | TechNet

Microsoft
Your potential. Our passion.™

© 2009 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. This information is for informational purposes only and represents the current state of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted as a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS PRESENTATION.



Microsoft | TechNet