Introduction to Entity Framework

Authored by : Amit Dhandal

Presented by : Pushkar Kulkarni

Agenda

- Traditional Data Access approaches
- ORM Basics
- Introduction to Microsoft Entity Framework

Traditional Data Access approaches

- Purpose of Data Access Layer:
  - Query data from data store
  - Data persistence
  - Track changes
- Data Access approaches used so far:
  - Resultset in classic ASP
  - ADO.Net and DataSet
  - DataReader\ DataAdapters

Traditional Data Access - Issues

- Issues with existing Data Access approaches:
  - Tabular Data Representation
  - Tight Coupling - DB schema and Business Logic
  - Loose-Typing - DataRow Cell Type ➔ Object
  - DataSet Performance

Using classes to Organize Data

- Class ➔ Table schema
- Class Instance ➔ Table Row\ Record

Advantages:

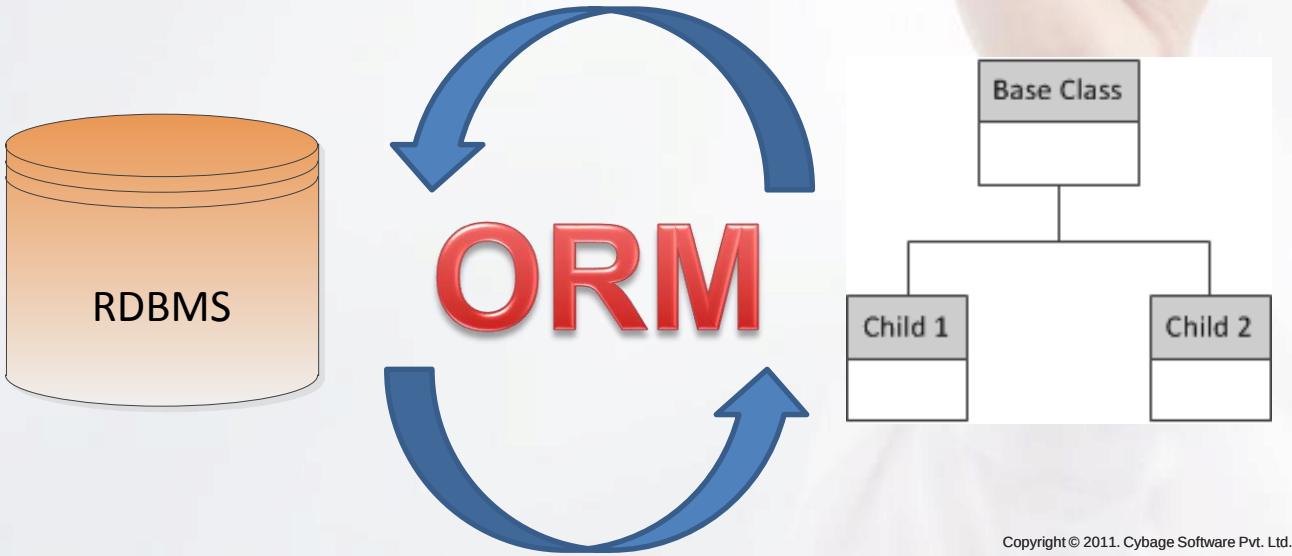
- Strong Typing
- Compile-time checking
- Ease of development
- Storage agnostic interface
- Self-Validation in Classes.

ORM Basics

Relational Model: Efficient storage and retrieval

Object Model: Real-world representation of data

Object/ Relational Mapping:



Advantages of ORM

- Productivity
- Maintainability
- Performance

.Net Entity Framework → ORM by Microsoft

- Part of .Net framework
- Integrated into Visual Studio
- Database type and version independent
- **Recommended for data access by Microsoft!**

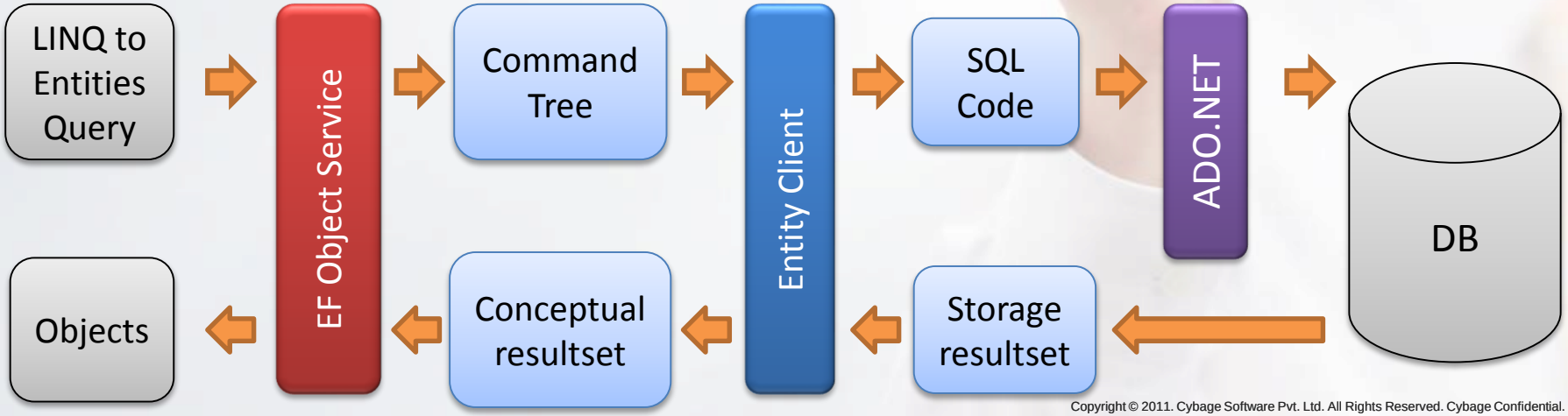
EF Demos – DB First

- Generating Model
- Editing Model
- SQL Query Generation
- Inheritance Model
- Entity Type and Complex Type
- Function Import

Querying the Object Model

Write queries against Entity Classes and NOT against actual Tables

Internal Flow:



Connecting to Database

Convention:

Entity Container name ➔ ObjectContext class name ➔ Connection String name

Setting up Connection String:

- metadata: Location of CSDL, SSDL and MSL file separated by "|"
- provider: ADO.NET provider name for underlying database.
  (ref: providerName attribute of generic Connection String)
- provider connection string: Actual connection string for underlying database.

Important: IQueryable VS IEnumerable

When is a query executed?

SELECT Query:

- *foreach* or *for* loop on result set
- Methods like ToList, ToArray, First, Single etc are called
- Lazy Loading > Navigation property accessed

INSERT, UPDATE, DELETE:

- *ObjectContext.SaveChanges() method*

.Net Entity Framework

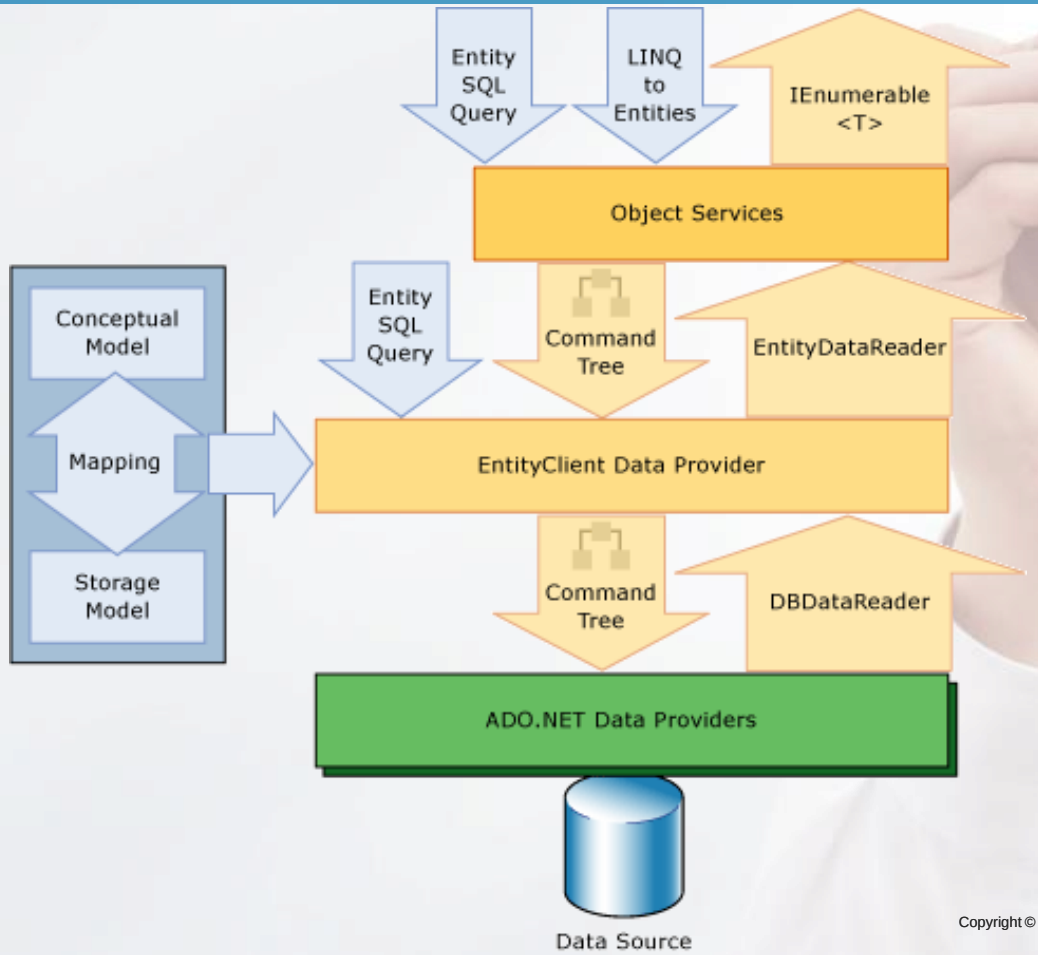
Entity Framework – Mapping files:

Filename	Description	Alternative name	Extension
Conceptual model	Describes the model classes and their relationships	Conceptual schema, conceptual side	CSDL
Storage model	Describes the database tables, views, and stored procedures, and their keys and relationships	Storage schema, storage side	SSDL
Mapping model	Maps the conceptual and storage models	Mapping schema, mapping side	MSL

.Net EF – Development Workflows

	Designer centric	Code centric
New Database	**Model first** • Create .edmx model in designer • Generate database from .edmx • Classes auto-generated from .edmx	**Code first** • Define classes and mapping in code • Database auto-created at runtime
Existing Database	**Database first** • Reverse engineer .edmx model • Classes auto-generated from .edmx	**Code first** • Define classes and mapping in code (Reverse engineer tools available)

EF Data Access Architecture



EF Demo

- Code First Approach

Install Code First support in Visual Studio

Step #1: Install “NuGet Package Manager” from

- Tools > Extension Manager
- www.nuget.org

Step #2: Install NuGet EntityFramework support using either:

a) Solution Explorer > right-click on References > “Manage NuGet Package” > Search and Install “EntityFramework”

b) Solution Explorer > right-click on References > “Add Library Package Reference” > Search and Install “EntityFramework”

c) Tools > Library Package Manager > Package Manager Console > type “install-package EntityFramework”

Verify that “EntityFramework.dll” is present in “References”

Develop app using Code First Approach

Methods to write Code First Entities:

- By Convention
  - Conventions to write POCO classes
- By Annotations
  - System.ComponentModel.DataAnnotations namespace:
    - Apply attributes/ annotations for changing mapping of POCO class and database table
- By Fluent API
  - Call EF API in OnModelCreation event handler.

DataAnnotations Values

Annotation	Purpose/Usage
Column	Column name associated with property
Table	Table mapped with class
Association	Indicate Foreign Key association, accepts key property names for both side entities
ConcurrencyCheck	Marks column for ConcurrencyMode = Fixed
DataType	DbType for database column
ForeignKey	Placed on Foreign Key property, takes name of related Navigation Property
Key	Indicates Property as Primary Key
MaxLength, MinLength	Min and Max Length for nvarchar column
Required	Indicates NOT NULL column
Timestamp	Indicates datatype for column as Timestamp\ rowversion

DbModelBuilder “Entity” APIs

Annotation	Purpose/Usage
HasKey	Configures Primary Key for this entity
HasMany	Configures Many relation “from” this entity
HasOptional	Optional relationship, entity can be saved without specifying this relation. FK is Nullable field.
HasRequired	Compulsory relationship, entity can NOT be saved without specifying this relation. FK is NOT NULL.
ToTable	Configures Table name mapped to this entity
Property	Returns specified Property of this entity
Ignore	Excludes Property from database mapping

DbModelBuilder “Property” APIs

Annotation	Purpose/Usage
HasColumnName	Configures Db Column Name mapped with this property
HasColumnOrder	Order of column in Composite key and while storing in Db
HasColumnType	Database type for this property
HasMaxLength	Specifies Max length for this property
IsConcurrencyToken	ConcurrencyMode option set for this property
IsOptional	Associated DB column is Nullable
IsRequired	Associated DB column is Not Nullable

DbModelBuilder “Relationship” APIs

HasOptional() – Returns OptionalNavigationProperty Configuration
HasRequired() – Returns RequiredNavigationPropertyConfiguration

Navigation Property API	Purpose/Usage
WithMany	Configures “1—0…*” [one-to-zero or many] relationship without having navigation property on other entity
WithOptional	Configures “1—0…1” [one-to-zero or one] relationship without navigation property on other entity
WithRequiredDependent	Configures “1—1..*” [one-to-many] relationship having THIS entity as dependent and target entity as Principal
WithRequiredPrincipal	Configures “1—1..*” [one-to-many] relationship having THIS entity as Principal and target entity as Dependent
Association > WillCascadeOnDelete	Configures Cascade Delete property for this relation

Links

- Impedance Mismatch
  http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch
- Entity Framework
  http://msdn.microsoft.com/en-in/data/ef.aspx
- Books
  http://msdn.microsoft.com/en-us/data/aa937716

Any Questions?

Amit Dhandal
Sr. Technical Architect
CT2-3-8502
amitdhan@cybage.com

Pushkar Kulkarni
Sr. Technical Architect
CT2-3-8504
pushkark@cybage.com

Thank you!