ASP.NET MVC 5.0

Lesson 5: Entity Framework 6
Features with Asp.Net Scaffolding

Lesson Objectives

- Why Entity Model
- What is Entity Framework
- Entity Framework Architecture
- Entity Framework Approaches
- Code First Approach in MVC

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

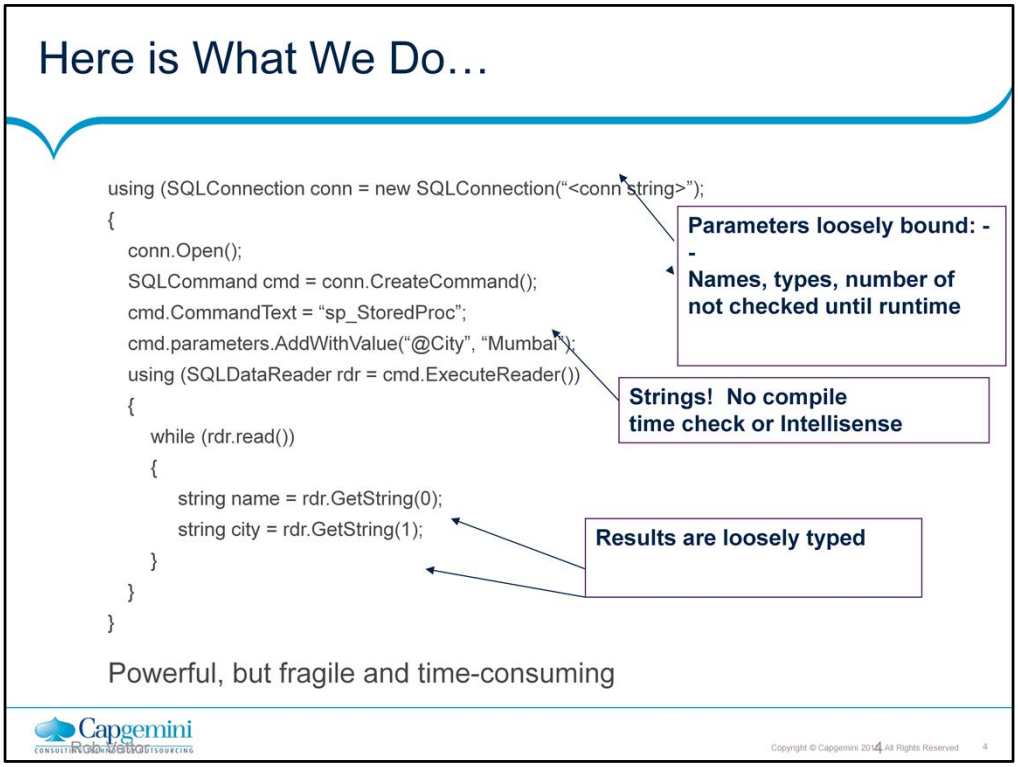
 2

Software Engineer's Life…

- We build Applications for different domains
- Each application talks to relational database
- Learn (at least) 2 different languages (C# and SQL)
  - Different syntax
  - Different type systems
  - Different tools
  - Different paradigms: Object vs. Procedural
- Also must learn the API that binds together: ADO.NET
- Powerful, but fragile and time-consuming

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3


Here is What We Do…
using (SQLConnection conn = new SQLConnection("<conn string>");
{
conn.Open();
SQLCommand cmd = conn.CreateCommand();
cmd.CommandText = "sp_StoredProc";
cmd.parameters.AddWithValue("@City", "Mumbai");
using (SQLDataReader rdr = cmd.ExecuteReader())
{
while (rdr.read())
{
string name = rdr.GetString(0);
string city = rdr.GetString(1);
}
}
}
Parameters loosely bound: --
Names, types, number of not checked until runtime
Strings! No compile time check or Intellisense
Results are loosely typed
Powerful, but fragile and time-consuming
Capgemini
R&D VS
Copyright © Capgemini 2014 All Rights Reserved
4

The Main Issue so far..

- Objects != Relational Data

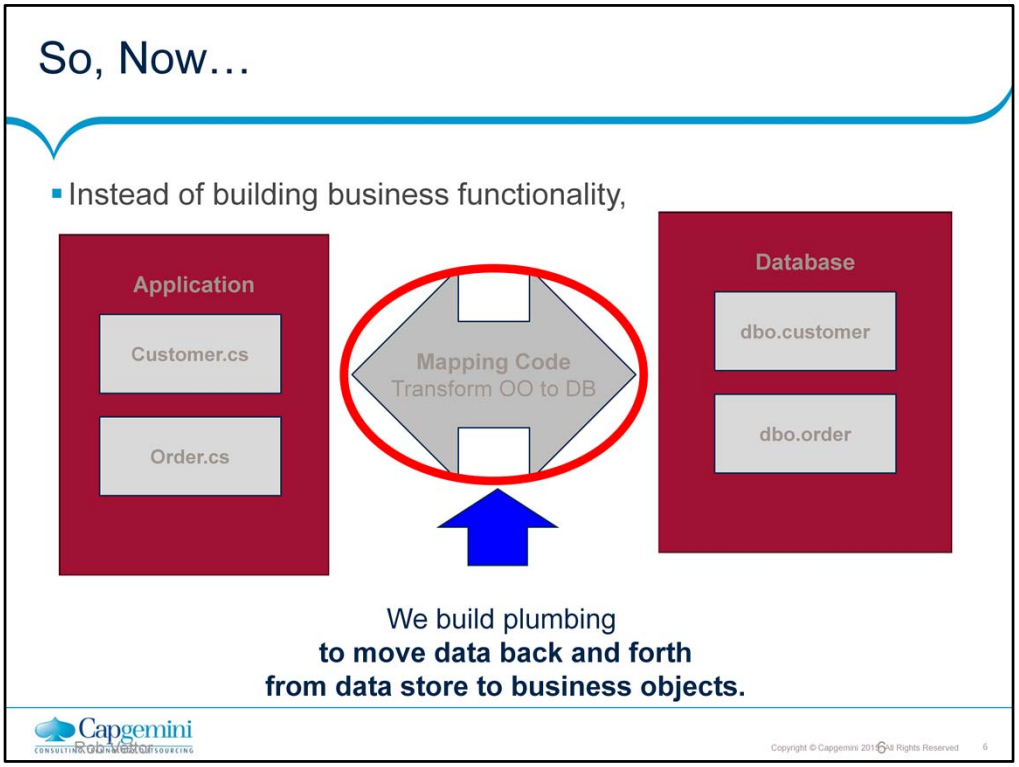
The “Impedance Mismatch”

Relational Database

Conceptual / Business Model (Objects)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

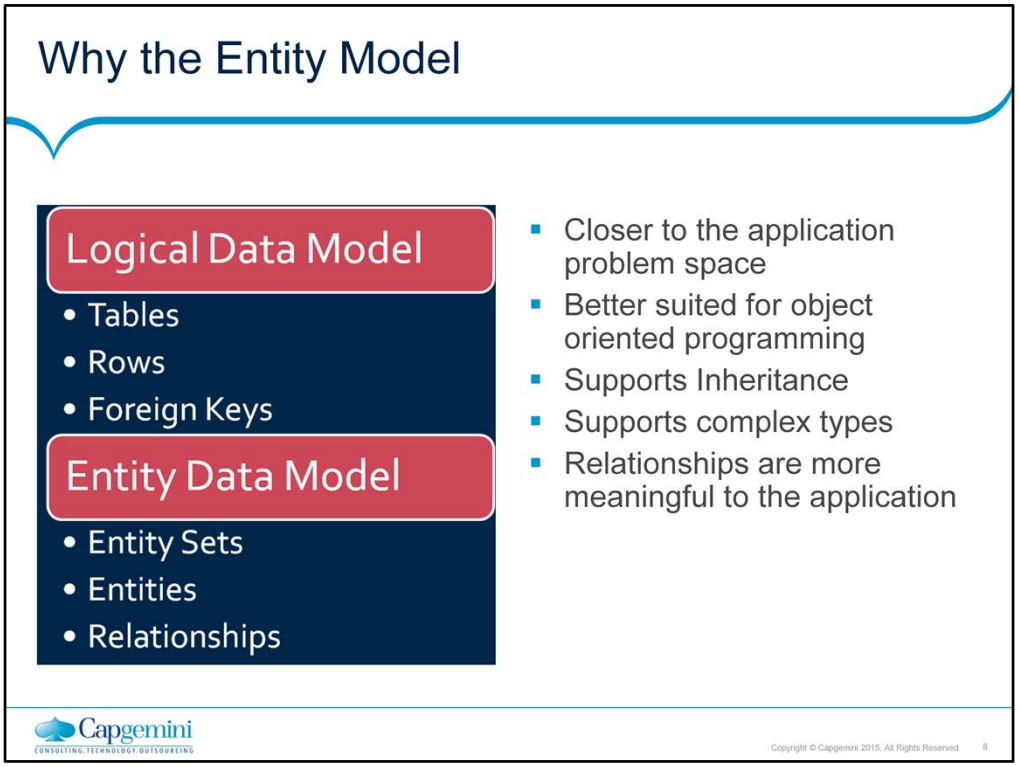
 5


So, Now…
Instead of building business functionality,
Application
Customer.cs
Order.cs
Mapping Code
Transform OO to DB
Database
dbo.customer
dbo.order
We build plumbing
to move data back and forth
from data store to business objects.
Capgemini
Copyright © Capgemini 2015 All Rights Reserved
6

Solution – Entity Framework

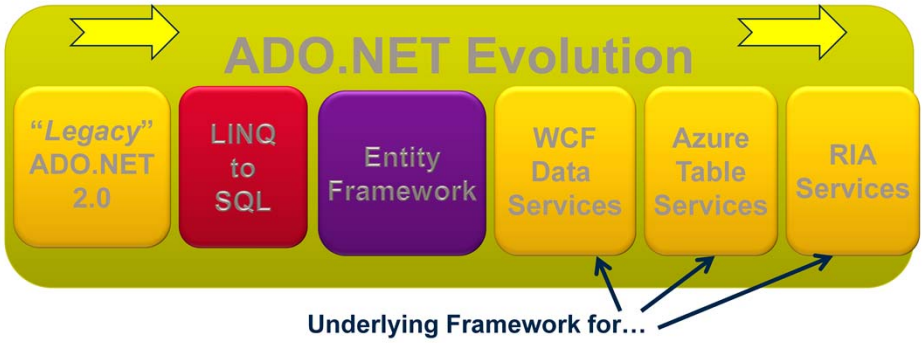
- Data access framework
- Supports data-centric applications and services
- Enables programming against a conceptual application model
- Enables independency of any data storage engine or relational schema

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING


Why the Entity Model
Logical Data Model
• Tables
• Rows
• Foreign Keys
Entity Data Model
• Entity Sets
• Entities
• Relationships
Closer to the application problem space
Better suited for object oriented programming
Supports Inheritance
Supports complex types
Relationships are more meaningful to the application
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 8

What is the Entity Framework?

- EF is a data access framework from Microsoft that helps bridge the gap between data structures and objects in your applications.



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Programming against a Model

- EF uses a model called an Entity Data Model (EDM)
- EDM is a client-side data model
- EDM is an abstraction layer on top of the data storage
- Remove the pain of
  - Interacting with the data storage
  - Translating the data into objects

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

What does it do?

- Develop against conceptual view of your data, instead of data store itself
  - Automatically generates strongly-typed entity objects that can be customized beyond 1-1 mapping
  - Automatically generates mapping/plumbing code
  - Automatically translates LINQ queries to database queries
  - Automatically materializes objects from data store calls
  - Automatically tracks changes, generating updates/inserts
  - Delivers variety of visual modeling tools

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

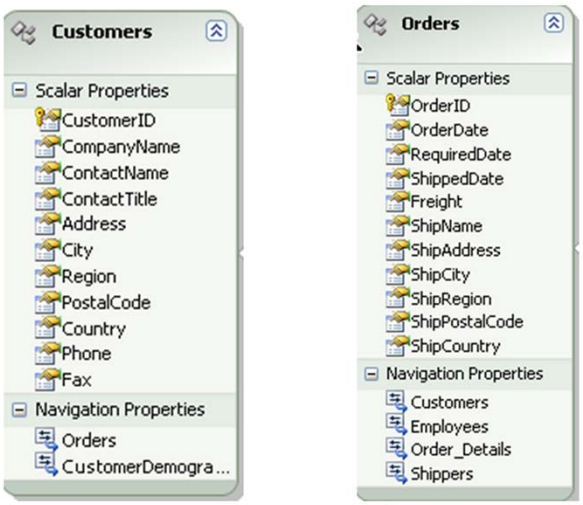
 11

The “Entity” in Entity Framework

- Items described in the EDM are called entities
- Entities have only properties but no behavior
- Entities can have relationships with other entities

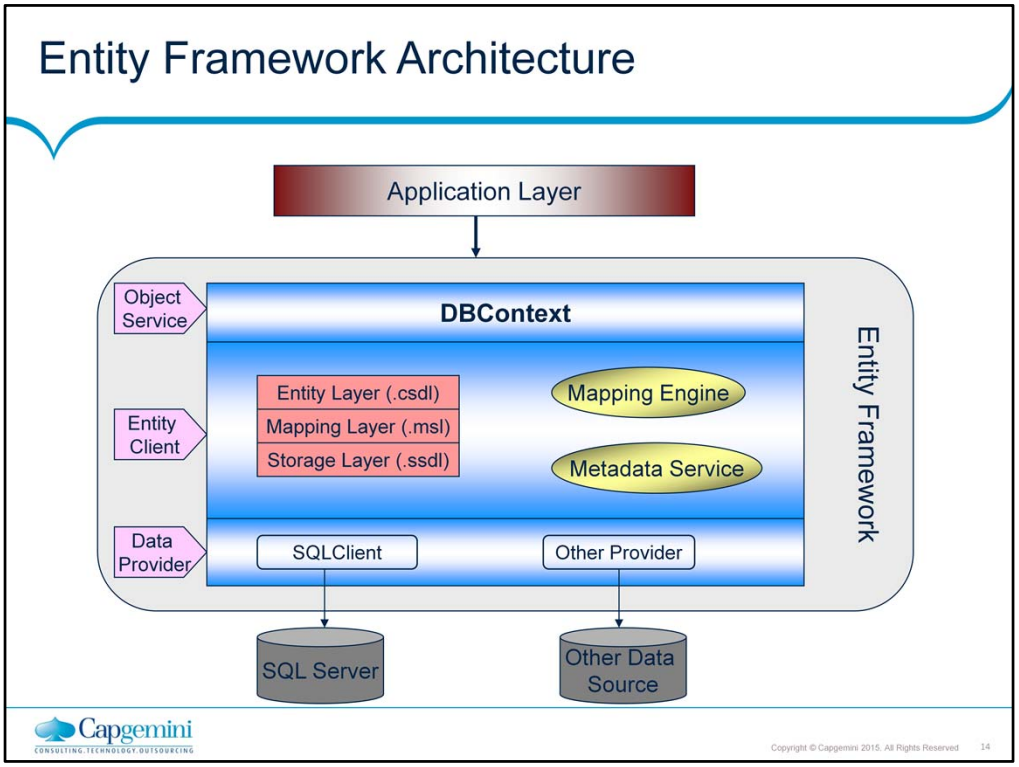
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Entity Framework's Entities



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13



Entity Framework Architecture

After having briefly taken a look at Entity data model, we will now get into the details and understand the Entity Framework Architecture.

The slide shows the architecture, It comprises following layers:

Data Provider: This is the lowest layer which translates the common SQL languages such as LINQ via command tree to native SQL expression and executes it against the specific DBMS system.

Entity Client: This layer exposes the entity layer to the upper layer. You can write code to query data using Entity SQL an entity aware language. The entity model is mapped to the database table via mapping specification language and mapping engine. In essence, the entity client provides the ability for developers to work against entities in the form or rows and columns using entity SQL queries without the need to generate classes to represent conceptual schema.

The Entity Client shows the entity framework layers which are the core functionality. These layers are called as Entity Data Model as mentioned earlier. The Storage Layer (.ssdl) contains the entire database schema in XML format.

The Entity Layer (.csdl) which is also an XML file defines the entities and relationships. Within the application you always work with Conceptual Layer. For this you can use: Entity SQL, LINQ to Entities (Another Language Integrated Query Facility), Object Services.

The Mapping layer (.msl) is an XML file that maps the entities and relationships defined at conceptual layer with actual relationships and tables defined at logical layer.

The Entity Framework uses all the three XML files to create, update, delete and read operations against entities and relationships. It also supports mapping entities to stored procedures in the data source.

The Metadata services which is also represented in Entity Client provides centralized API to access metadata stored in the .csdl, .ssdl, .msl.

Object Services: This is the ORM layer of Entity Framework. It represents the data result to the object instances of entities. This services provides rich ORM features like primary key mapping , change tracking etc… Both LINQ and Entity SQL could be used to write query. Within the Object Services layer is the ObjectContext which represents the session of interaction between the applications and the data source. You will primarily use the ObjectContext to query, add, delete instances of entities and to save the changed state back to the database.

DBContext

- The DBContext class provides facilities for querying and working with entity data as objects
- It also keeps track of the entity objects, along with the state information (added, modified, deleted)
- It has a ability to update entities and write changes back to the database.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

The three parts of Entity Data Model

- SSDL: Storage Schema Definition Language, also known as the conceptual model
  - the SSDL (Storage Model) section is an XML schematic representation of the mapped data store
- CSDL: Conceptual Schema Definition Language, also known as the storage model
  - it is a conceptual design template for the object model that applications will use to build their applications against

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

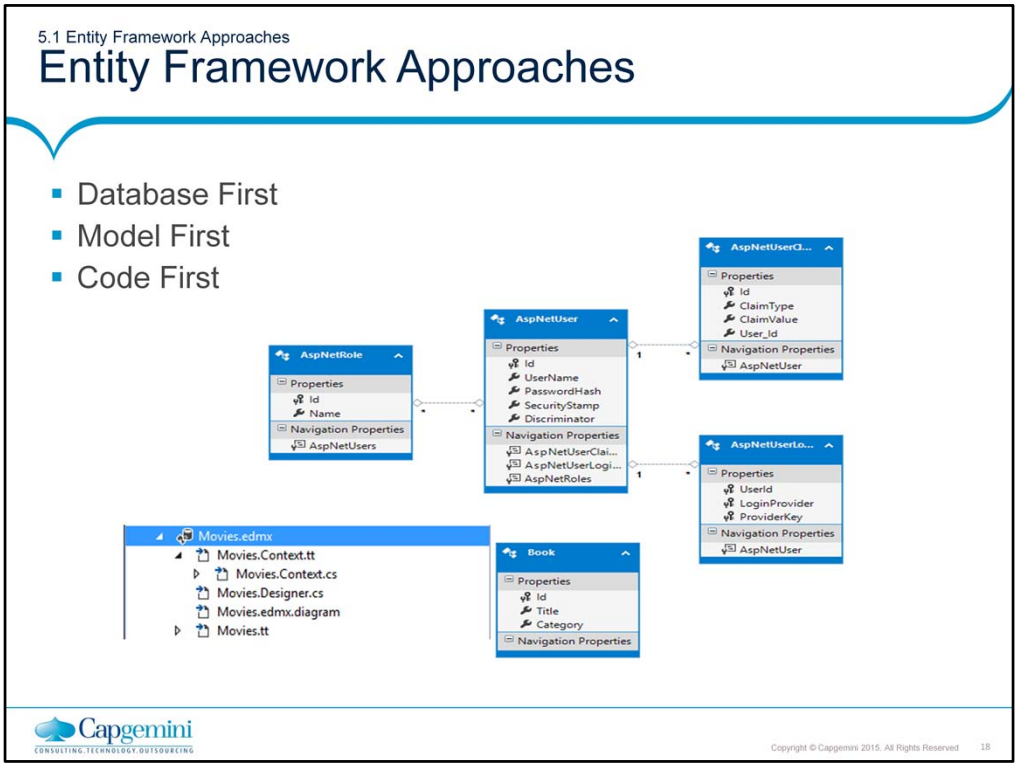
 16

The three parts of Entity Data Model

- CS (MSL): Mapping information
  - The mapping section is a specification that is used to connect the CSDL types to the database metadata defined in the SSDL
  - The mapping is a layer that resides between the conceptual and store layers.
  - Its entire purpose is to provide the mapping between entities (entity properties) to the tables and columns in the data store

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17



There are three approaches to working with EF. You can do Database first, Model first and code first.

Database First:

With the database first approach, you can go to your project and add an item of type entity framework data model, point it to a database, and EF will reverse engineer the database into a model, provide you a design service where you can tweak the model and keep the model updated with the database

The database first approach will generate the code that you need to access the database by adding T4 templates to the project. Those are the files with the .tt extension. Those files read the model at compile time and generate C# code. You'll find the C# code files nested behind the tt file in Solution Explorer.

Model First:

A model first approach is very similar because you use the same design service, but you typically start with no database and use the designer to generate SQL scripts that can create the database once you've added the entities that you want on to the design surface.

Code First:

The code first approach is where we write some classes that represent the entities that we want to store in the database and the Entity Framework will make that happen. No design services, just C# code.

But regardless of which approach that you use, you'll always be using the Entity Framework DB context API.

That's the go-to API since the last version of EF shipped, EF version 5. Even the database first approach generates DB context code and it is the DB context API that we'll use in our course . In fact, we'll use DB context classes. Both of them will work with the same database instance. Both will live in the same project and both will have some migration scripts to keep the database in sync with the code.

DEMO

- Demo of Code First approach in MVC

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

Summary

- In this lesson we discussed about the following:
- Why Entity Model is useful
- What is Entity Framework
- Entity Framework Architecture
- Entity Framework Approaches
- Code First Approach in MVC

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20