Server side Web Development

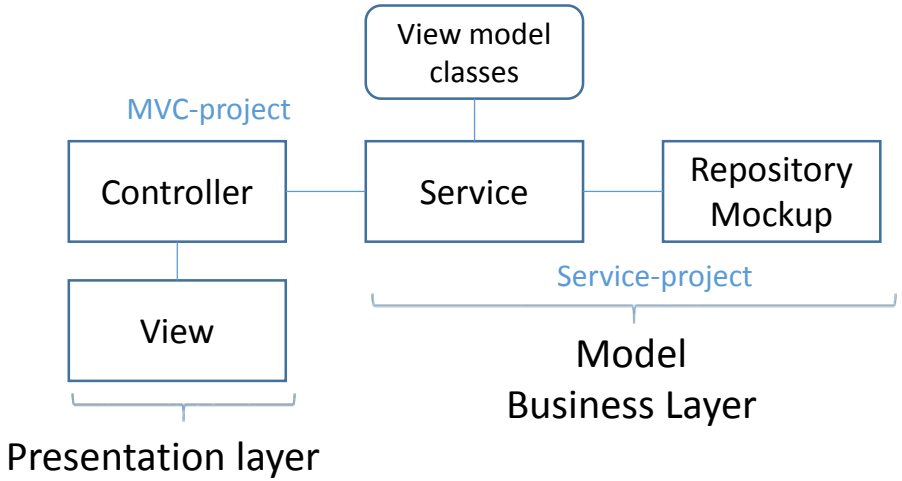
Lecture 5

Data access using ORM

Outline

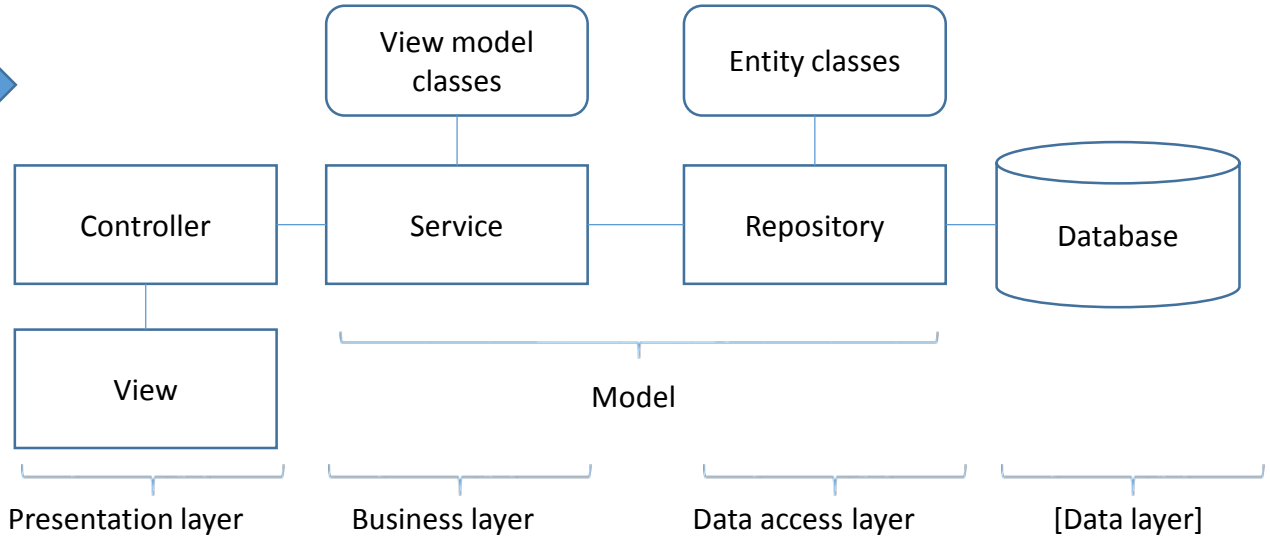
- Repository and database access
  - Using Entity Framework as ORM

What to acquire



Starting with two layers using a mockup

Ending up with three layers
Using Entity Framework as ORM
For the data access

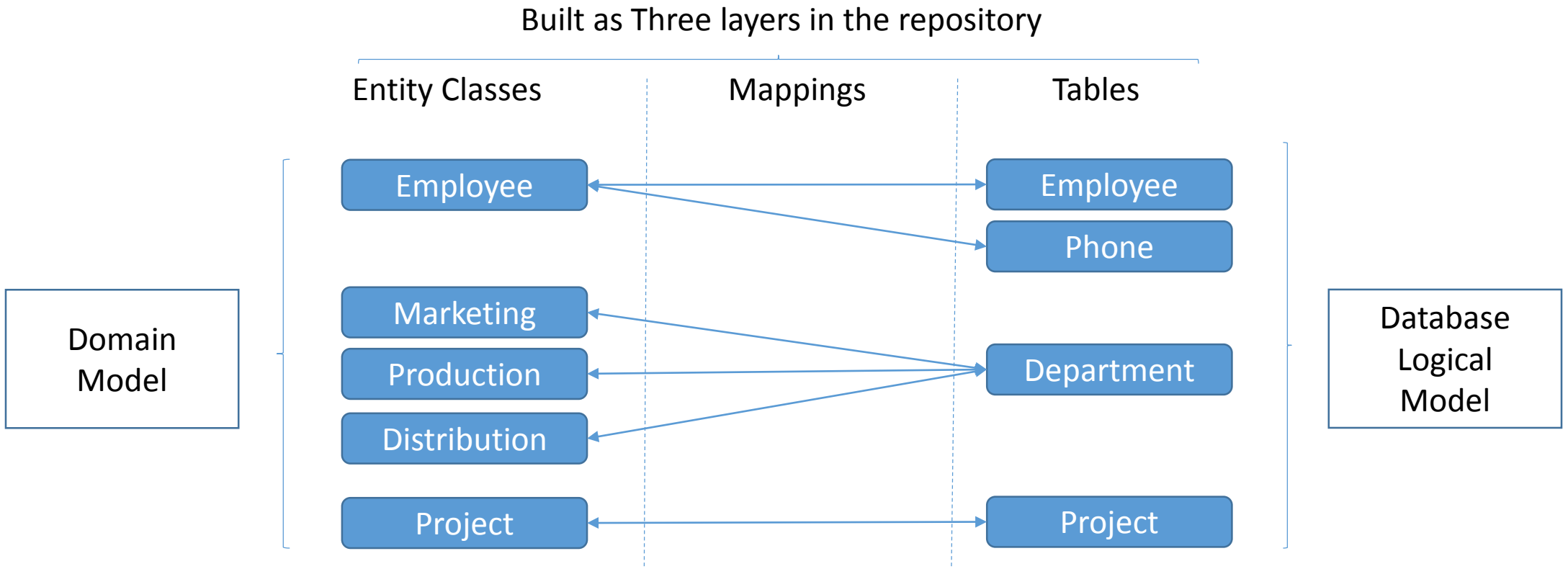


Ways of using Entity Framework

- Different ways to work with Entity Framework (EF):
  - Dabase first
    - Through visual studio the EF designer will use an already existing database to generate Entity classes and code for managing data to/from the database
    - Manual changes of the database is supported
  - Model first
    - The EDM designer tool in Visual Studio is used to model the database
    - EF uses the model to create the database
  - Code first
    - The domain model classes and datamappings are defined first
    - EF uses Model Classes and database mappings to create the database

Entity Data Model

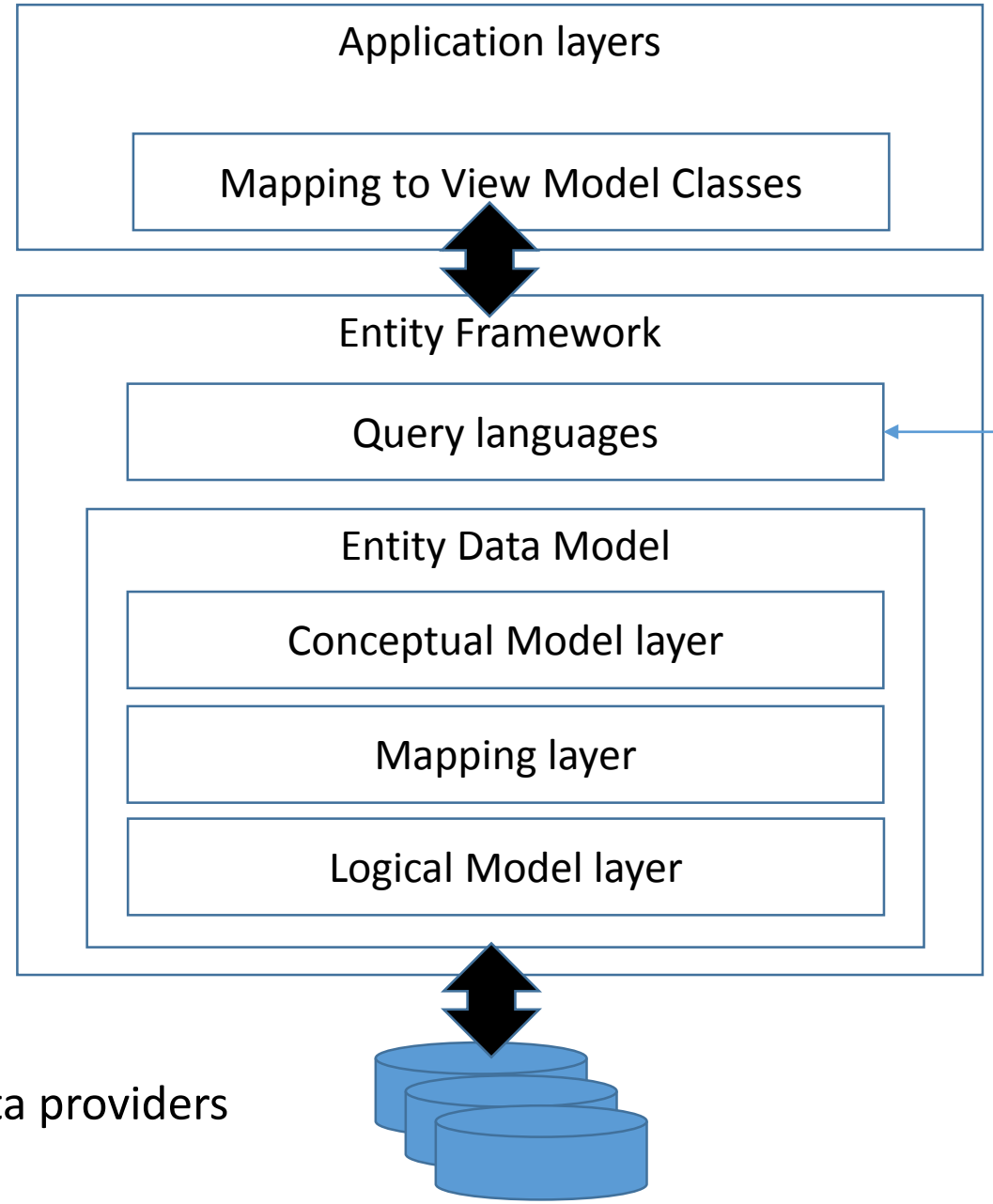
- EF uses a model concept called Entity Data Model (EDM)
  - There is a mapping issue between the Domain Model and the Logical model of the database (Relational database in most cases)



The three layers in EDM

- Conceptual layer
  - Contains the Entity Classes
- The Mapping Layer
  - Map Logical to Conceptual
- The Logical Model Layer
  - Tables, Stored Procedures, Views etc…

Written in XML-files



Linq to SQL
Entity SQL
DBContext (Wrapper of Objectcontext)

Hands on (1)…

- Add a “Repository” project to the solution
- Adding EF
  - Right click repository project and select “Manage NuGet Packages”
  - When loaded select “Entity Framework” and click Install
- Adding an Entity Data Model for the data provider
  - Right click repository project and select “Add” -> “New Item”
  - Select “Data” in the left pane
  - Select “ADO.NET Entity Data Model” in the central pane
  - Name the .edmx – file with an appropriate name for the project
  - In next window, choose “Generate from database” if you intend to use an already existing database

…Hands on (2)…

- Adding an Entity Data Model for the data provider, cont.
  - Set up the connection properties
    - Data source: “Microsoft SQL Server (sqlclient)”
    - Select type of authentication, SQL-server will require userid and password
    - Enter the server name and database name
  - Import the required schema from the database
    - Check required tables
    - Check “Pluralize or singularize generated object names”
    - Check “Include foreign key columns in the model”
  - Click “Finish”
  - Will bring up the EDM with a graphical view of the generated Entity classes
- Build the solution (ctrl+shift+B)
  - The files with extension .tt contain T4 code which generates the Entity class files

…Hands on (3)…

- Add a reference from the Service project to the Repository project
- Copy the connection string from app.config file in the Repository project to the Web.config file in the Web-project
- Adding query support
  - Add a folder to the Repository project for defining CRUD queries
  - Add a class for each entity class to define queries for CRUD operations
    - List()
    - Read()
    - Add()
    - Update()
    - Delete()
    - Add using statement “`using System.Data.Entity;`” (for the Update – method)

…Hands on (4)…

- Add using statement “using Repository.Support”
  - In order for the methods in the Service project to know about the repository support for the data base access
  - (Instead of the using statement for the Mockup)
- The data objects from the repository are generated from Entity classes, which are not the same as the View model classes. Therefore they need to be translated.
  - The mapping is in most cases quite straight forward, however rather cumbersome to write

…Hands on (5)…

- The following needs also to be added to Repository project:

The name o the class and its constructor should be the name of the database + “Entities”

You have to change it manually to public partial

```
namespace Repository
{
    public partial class dbprojectengEntities
    {
        static dbprojectengEntities()
        {
            var foo = System.Data.Entity.SqlServer.SqlProviderServices.Instance;
        }
    }
}
```

“A quirk that I have not got to the bottom of yet is that Copy Local is not enough to ensure that Entity Framework 6 can find the Sql Provider. The final step was to reference a type in the base domain context (which overrides the databasecontext you get from EF). This is echoed elsewhere, on this Stack Overflow answer.”
http://robsneuron.blogspot.se/2013/11/entity-framework-upgrade-to-6.html

Reading

- Books:
  - Kanjilal, Joydip “Entity Framework Tutorial - Second Edition”, Fulltext biblioteket
  - Driscoll, Brian ” Entity Framework 6 Recipes, Second Edition”, Fulltext biblioteket
  - Lerman, Julia “Programming Entity Framework: DbContext”, Fulltext biblioteket
- Links:
  - https://msdn.microsoft.com/en-us/library/jj729737(v=vs.113).aspx
  - https://msdn.microsoft.com/en-us/magazine/hh148150.aspx
  - http://thedatafarm.com/
  - https://www.tutorialspoint.com/entity_framework/entity_framework_relationships.htm