



Service Web .Net

Cours n°4 : Architecture & Data

Pierre-Loïc CHEVILLOT – Sébastien BEREIZIAT

Pierre-loic.chevillot@capgemini.com – sebastien.bereiziat@capgemini.com



Data Access

Base de données

- DBMS : DataBase Management Service (SGBD : Système de gestion de base de données)
 - Oracle
 - MySQL
 - MSSQL Server
 - CosmosDB
- Relationnelle (SQL) ≠ NoSQL (Not Only SQL)
 - MongoDB
 - CouchDB / Couchbase
 - CosmosDB

NoSQL

- NoSQL = Not Only SQL
- Paradigme basé sur les données basée sur un modèle d'infrastructure matérielle.
- Emergence dans les années 2010 suite à l'arrivée des « Datalake »
 - Cluster de données
 - Agrégats de données sur différents serveurs
- Données stockées sous forme
 - Agrégats : un agrégat sont des informations qui interagissent peu entre elles, et dans lesquelles on peut faire une requête non prévue dans les cas d'utilisation immédiats.
 - Graphes : stockage d'information sous la forme de graphe. Faciliter la plupart des requêtes en supprimant les jointures
 - Sans-Schéma: Stockage de données hétérogènes au fur et à mesure de leur alimentation. Grande flexibilité et capacité d'adaptation, logique de l'application plus complexe pour intégrer les données de cette base.

Relationnelle

- Information organisée dans des tableaux a 2 dimensions (Table)
- Base de données => plusieurs relations entre les tables
- Langage des bases : SQL

ORM

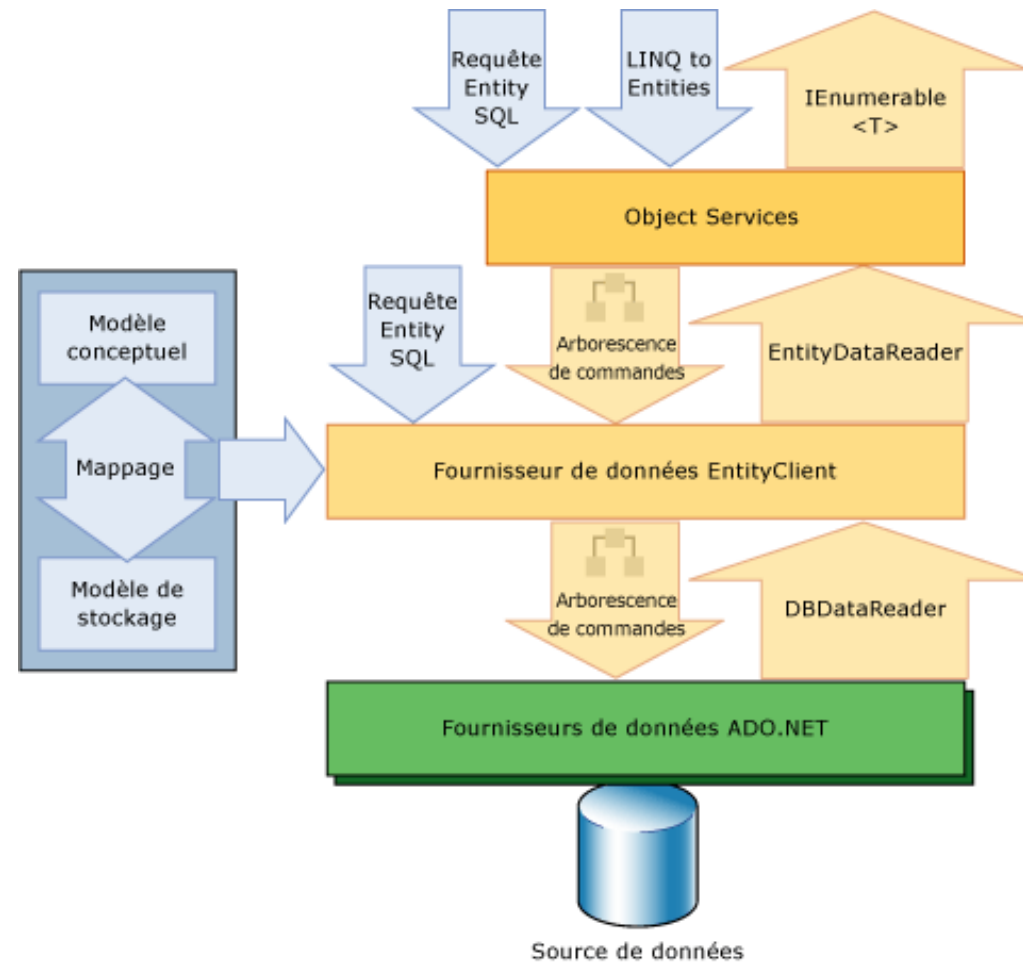
- Object-Relational Mapping
- Interface entre une application et une base de données
- But : Simuler la base de données orientée objet
- Mappe les correspondances entre la base de données et les classes de l'application.
- Requetage dans une BDD sans écrire une seule ligne de SQL
- -- : Couche logicielle supplémentaire qui peut jouer sur les performances, la maintenabilité. Certains ORM ont leurs limites, exemple la suppression en bloc de données.
- ++ : quantité de code écrite, homogénéité avec le langage objet.

ORM

- Java
 - Hibernate
 - JAVA Persistence API
- .Net
 - LinqToSQL
 - NHibernate
 - Entity Framework
 - EFCore
- Ruby
 - Active Record
- PHP
 - Doctrine
- Python
 - SQLAlchemy
- Node.js
 - sequelize

Entity Framework

- Développé depuis 2008 (VS2008 .NetFW3.5)
- Embarque un moteur d'abstraction et de représentation graphique de la base de données
- Autogénération du mappage base de donnée / entités objets
- Plusieurs fournisseurs de base de données (SQLServer, Oracle, MySQL, SQLLite)
- Uniquement WindowsOS



Entity Framework



Le langage CSDL (Conceptual Schema Definition Language) définit le modèle conceptuel. Le langage CSDL est l'implémentation de la Entity Framework du Entity Data Model.



Le langage SSDL (Store Schema Definition Language) définit le modèle de stockage, également appelé « modèle logique ».




Le langage MSL (Mapping Specification Language) définit les mappages entre le modèle de stockage et le modèle conceptuel.



Le Fichier EDMX (Entity Data Model) en format XML qui contient le fichier csdl et ssdl, et les mappages entre les 2 (msl)

EF Core

- Open-Source, lightweight (système de Nuget) , extensible
- Multiplateforme
- Plusieurs fournisseurs de base de données (SQLServer, Oracle, MySQL, SQLite)
- Release en 2016
- Autogénération de code uniquement sur la base de donnée. (Pas de fichiers inutile)



Création et utilisation d'un modèle de données

Database First

- Extraire les informations d'une base de donnée existante en les transformant en objet.
- Pas d'update (migrations). Cela écrase intégralement l'ancien mappage de la base
- Uniquement EF
- Créer l'EDMX depuis la base de donnée renseignée

Code-First

- Création d'une nouvelle base de données OU utilisation d'une base de données existante
- Créer l'ensemble des fichiers de classes Entités et le dbContext associées
- Enregistrement de chaque « modification » de la base dans une Migration
 - Montante : Modification de la base a effectuée
 - Descendante : Comment revenir a l'état précédent
- Ligne de commande : add-migration, update-database, db-scaffold
- Scaffolding remplace l'intégralité du dbContext
- Migrations sont écrite en C#, grâce au Fluent API.

Fluent API

- Ensemble de méthodes, points d'accès sur le framework EFCore
- Gestion par le code du mappage « C# » / « SQL »
 - C# : dans le dbContext
 - Annotations : dans le model directement
- Manipuler la gestion des Tables, Colonnes, Clés, etc...
- Permet de créer une seed directement a la création de votre base de données.

DbContext

- Classe liant « Entités » et base de données

```
using Microsoft.EntityFrameworkCore;

namespace EFModeling.FluentAPI.Required
{
    class MyContext : DbContext
    {
        public DbSet<Blog> Blogs { get; set; }

        #region Required
        protected override void OnModelCreating(ModelBuilder
modelBuilder)
        {
            modelBuilder.Entity<Blog>()
                .Property(b => b.Url)
                .IsRequired();
        }
        #endregion
    }

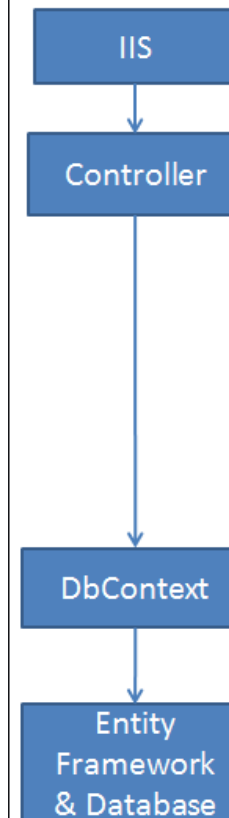
    public class Blog
    {
        public int BlogId { get; set; }
        public string Url { get; set; }
    }
}
```

Unit of Work Pattern

- SoC : Separation of Concern
 - Séparation net des composants selon leurs fonctionnalités
- Repository
 - Agrégation des méthodes d'accès a une base triée par métier / technique
 - Capable d'être mocké / testé
- DbContext
 - Uniquement le mappage DBMS / Objet

No Repository

Direct access to database context from controller.



With Repository

Abstraction layer between controller and database context. Unit tests can use a custom persistence layer to facilitate testing.

