ORM

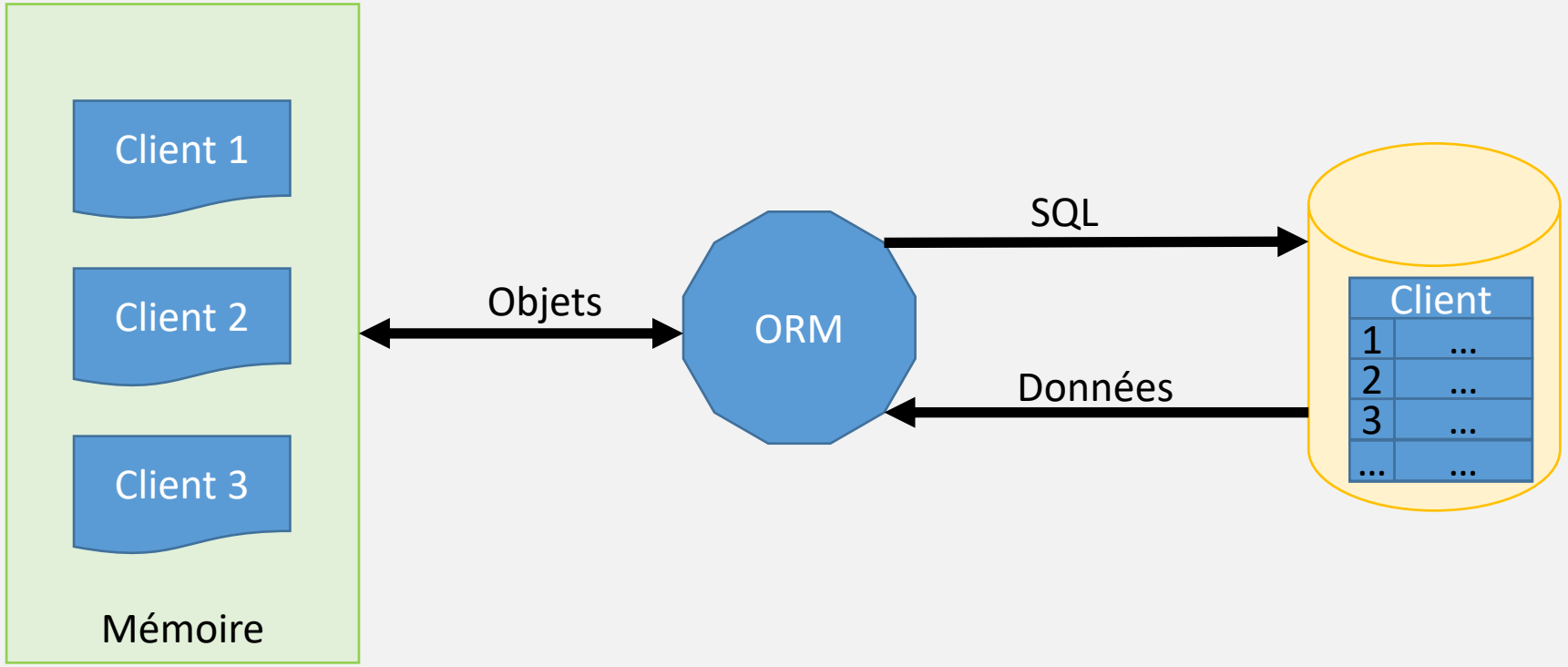
Photo : https://unsplash.com

Objectifs

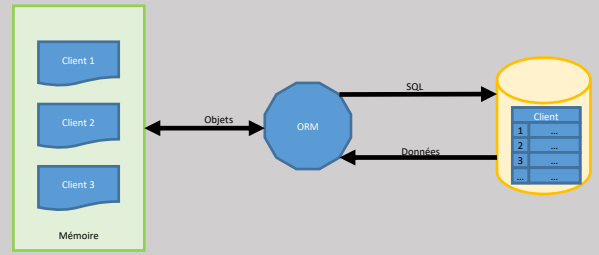
- Généralités sur les ORMs
- Entity Framework

Généralités – ORM (Object Relationnal Mapping)

- Fait le lien entre les classes de votre code et des tables
- Chaque objet correspond à un enregistrement
- Il s'occupe de convertir les actions du développeur en requêtes SQL

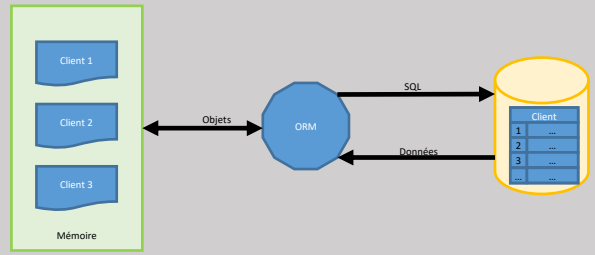


Entity Framework



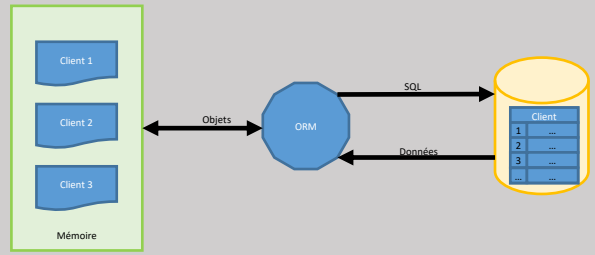
- ORM open source de Microsoft
- Permet d'interragir avec plusieurs SGBD (SQL Server, MySQL/MariaDB, Oracle, SQLLite, etc.) avec sensiblement le même code
- Fonctionne par conventions :
  - Le nom de la classe correspond au nom de la table (Ex. : classe Client <-> table client)
  - Chaque nom de propriété C# (get; set;) correspond au nom d'une colonne dans la table
  - Le nom de la clef primaire est « <nomClasse>Id » (Ex. : ClientId)
  - Les noms des clefs étrangères peuvent être :
    - <nomPropriété>
- Ces conventions peuvent être remplacées par des annotations (Attributs en C#) sur les classes et propriétés
- On peut aussi utiliser une approche de description dite « fluent »

Entity Framework



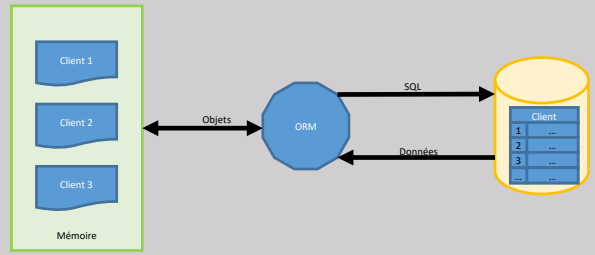
- Il existe deux approches
  - Code First : la base de données est créée par EF à partir de ce qui a été configuré
  - Database First : il faut créer la configuration pour se coller à la base de données
- Nous allons utiliser l'approche DB First
- EF utilise massivement Linq pour les requêtes

Entity Framework



- Une classe correspond à la description d'un type d'enregistrement et donc à une table
- Chaque propriété correspond à une colonne
- Les annotations permettent de définir des contraintes unitaires sur les colonnes ou de configurer EF. Exemple :
  - Key : Identifiant unique de l'objet
  - DataType : Type de la donnée (Exemple date)
  - Required : Le champ devient obligatoire
- Ces informations sont décrites dans l'espace de nommage : « System.ComponentModel.DataAnnotations » (https://learn.microsoft.com/fr-fr/dotnet/api/system.componentmodel.dataannotations?redirectedfrom=MSDN&view=net-7.0)
- Le contexte de base de données fait le lien entre une classe et une table
- Le contexte permet aussi de se connecter à la base de données

Entity Framework



- Généralement :
  - Il faut créer une classe par table et une propriété par champ
  - Il faut créer une classe qui hérite de la classe « DbContext » afin de configurer l'accès vers la base de données :
    - Il faut spécifier le fournisseur spécifique à la base de données utilisé avec Use<NomSGBD> (Ex. : UseSqlServer, UseOrable, UserMySQL)
  - **Une instance de DbContext est à créer par unité de travail, c'est-à-dire par tâche / transaction**
  - Il est conseillé d'utiliser une approche de type « Dépôt » afin de protéger votre code (Attention : c'est un anti-pattern mais nous allons quand même l'utiliser comme cela ici)
- Pour SQL Server :
  - Package nuget « Microsoft.EntityFrameworkCore.SqlServer »
  - Exemple de chaine de connextion :
    - « Server=.;Database=Module10_ORM;Trusted_Connection=True;MultipleActiveResultSets=true »
    - « . » = serveur local, instance par défaut
- Pour Oracle :
  - Package nuget « Oracle.EntityFrameworkCore »
  - Exemple de chaine de connexion :
    - `"Data source=localhost:1521/orcl; User Id=pfleon; Password=Bonjour01.+;"`

Création des entités - exemple

```
public class Personne
{
    public int PersonneId { get; set; } = 0;
    public string Nom { get; set; } = "";
    public string Prenom { get; set; } = "";

    public int? AdresseActuelleId { get; set; }

    [ForeignKey("AdresseActuelleId")]
    public Adresse? AdresseActuelle { get; set; }
    public List<Adresse>? Adresses { get; set; }

    […]
}
```

```
public class Adresse
{
    public int AdresseId { get; set; } = 0;
    public int PersonneId { get; set; } = 0;
    public string NoCivique { get; set; } = "";
    public string Odonyme { get; set; } = "";
    public string Ville { get; set; } = "";

[…]
}
```

Description d'un modèle de données - exemple

```
public class ApplicationDBContext : DbContext<ApplicationDBContext>
{
    private IDbContextTransaction? m_transaction;
    public ApplicationDBContext(DbContextOptions dbContextOptions) : base(dbContextOptions)
    { ; }

    public DbSet<Personne> Personne => Set<Personne>(); // { get; set; }
    public DbSet<Adresse> Adresse => Set<Adresse>(); // { get; set; }

    public List<Adresse> ObtenirAdressesPourVilleContenant(string p_partieNomVille)
    {
        return this.Adresse
                   .FromSqlRaw("EXECUTE Obtenir_Adresses_Ville_Contenant {0};", p_partieNomVille)
                   .ToList();
    }
}
```

Création du contexte exemple

```
IConfigurationRoot configuration =
        new ConfigurationBuilder()
                .SetBasePath(Directory.GetParent(AppContext.BaseDirectory)!.FullName)
                .AddJsonFile("appsettings.json", false)
                .Build();

_dbContextOptions = new DbContextOptionsBuilder<ApplicationDBContext>()
                .UseSqlServer(configuration.GetConnectionString("PersonnesConnection"))
                .UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking)
#if DEBUG
                .LogTo(message => Debug.WriteLine(message), LogLevel.Information)
                .EnableSensitiveDataLogging()
#endif
                .Options;

ApplicationDBContext dbContext = new ApplicationDBContext(_dbContextOptions);
```

Création DBContext avec un singleton

```
public static class DALDbContextGeneration
{
     private static DbContextOptions<ApplicationDBContext> _dbContextOptions;
     private static PooledDbContextFactory<ApplicationDBContext> _pooledDbContextFactory;
     static DALDbContextGeneration()
     {
          IConfigurationRoot configuration = new ConfigurationBuilder()
                    .SetBasePath(Directory.GetParent(AppContext.BaseDirectory)!.FullName)
                    .AddJsonFile("appsettings.json", false)
                    .Build();

          _dbContextOptions = new DbContextOptionsBuilder<ApplicationDBContext>()
                    .UseSqlServer(configuration.GetConnectionString("PersonnesConnection"))
                    .UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking)
                    #if DEBUG
                    .LogTo(message => Debug.WriteLine(message), LogLevel.Information)
                    .EnableSensitiveDataLogging()
                    #endif
                    .Options;

          _pooledDbContextFactory = new PooledDbContextFactory<ApplicationDBContext>(_dbContextOptions);
     }

     public static ApplicationDBContext ObtenirApplicationDBContext()
     {
          return _pooledDbContextFactory.CreateDbContext();
     }
}
```

Exemple de requêtes – Select

```
public List<Entite.Personne> ObtenirPersonnes(bool inclureAdresse = false)
{
    IQueryable<Personne> requete = this.m_dbContext.Personne;
    if (inclureAdresse)
    {
        requete = requete.Include(p => p.AdresseActuelle).Include(p => p.Adresses);
    }
    return requete.Select(p => p.VersEntite()).ToList();
}
```

Exemple de requête – INSERT

```
public void AjouterPersonne(Entite.Personne p_personne)
{
    Personne p = new Personne(p_personne);
    p.PersonneId = 0;

    this.m_dbContext.Add(p);
    this.m_dbContext.SaveChanges();
    this.m_dbContext.ChangeTracker.Clear();

    p_personne.PersonneId = p.PersonneId;
}
```

Exemple de requête – UPDATE

```
public void MAJPersonne(Entite.Personne p_personne)
{
    Personne p = new Personne(p_personne);

    this.m_dbContext.Update(p);
    this.m_dbContext.SaveChanges();
    this.m_dbContext.ChangeTracker.Clear();
}
```

Références

- Entity Framework core : https://docs.microsoft.com/en-us/ef/core/
- Configuration du contexte : https://docs.microsoft.com/en-us/ef/core/dbcontext-configuration/
- Fournisseurs BD :
  - https://docs.microsoft.com/en-us/ef/core/dbcontext-configuration/#dbcontextoptions : Liste incomplète mais plus détaillée
  - https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli
- Optimisation mémoire - Context pooling (regardez « Without dependency injection pour ce cours car on n'utilise pas ASP.Net Core app) : https://docs.microsoft.com/en-us/ef/core/performance/advanced-performance-topics