Les etapes :

1- n3aby Domaine b les classe w si famma classe numeration

2- les relation simple kima lexempel

3- les annotation lkol

4- fil config ngedo classe ili de5la b3atha

5- n3aytoulhom fil class DBContext

6- si fama tostring wala haja haka

7- add-migration databaseExam

8- update-Database

-------------------------------------class----------------------------------------

Les classes n7otouhom fel domaine

1-Type de classe detenu :

public class Customer

{

public int Id { get; set; }

public string Name { get; set; }

public Address Address { get; set; } // Propriété détenue

}

1ere methode de config en utilisant notation

[Owned] // Attribut indiquant un type d'entité détenu

public class Address

{

public string Street { get; set; }

public string City { get; set; }

public string PostalCode { get; set; }

}

2eme methode de config ki y9olek en utilisant fluent API

Entity Framework Core reconnaît automatiquement un type d'entité détenu grâce à l’attribut [Owned]. Sinon, vous pouvez configurer ce type explicitement dans le contexte :

public class AppDbContext : DbContext

{

public DbSet<Customer> Customers { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Customer>()

.OwnsOne(c => c.Address); // Configuration explicite

}

}

2-Type de classe enum

public enum PlaneType

{

BOING=1,AIRBUS=2

}

3-classe 3adeya

public class Flight

{ prop ...

public string Nom { get; set; }

}

--------------------------------------ANNOTATIONS-----------------------

-Obligatoire: [Required]

-obligatoire avec message d'erreur: [Required(ErrorMessage ="Champs obligatoire")]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*base\_de\_donné\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-ne voulons pas créer de colonnes correspondantes dans la base de données :[NotMapped]

-Nom d'une table dans une table db : [Table("MyFlight")]

-Nom d'une colonne dans une table db : [Column("Name")]

-Type de données d'une colonne :[Column("DoB", TypeName="DateTime2")]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Clé\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-clé primaire: [Key]

-commençant par l'index zéro : [Column(Order=1)]

-cle etrangere : [ForeignKey(“StandardId”)]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*un champ de 7 caractères\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

· L'expression régulière ^\d{8}$ correspond à une chaîne qui contient exactement 8 chiffres (\d représente un chiffre).

· Le ^ indique le début de la chaîne et le $ la fin de la chaîne.

==>[RegularExpression(@"^\d{8}$", ErrorMessage = "Le CIN doit contenir uniquement des chiffres.")]

-[MinLength(7, ErrorMessage = "MinLength 7")]

-[MaxLength(7, ErrorMessage = "MaxLength 7")]

ou

-[StringLength(7, MinimumLength = 7, ErrorMessage = "il faut etre 7 characters .. ... ...")]

-[StringLength(25, MinimumLength = 3, ErrorMessage = "First entre 3 characters et 25 characters .")]

ou

-[Range(0,int.MaxValue,ErrorMessage ="entier positive")]

-[Range(1,120, ErrorMessage="Age must be between 1-120 in years.")]

-positif ne depasse pas 24 : [Range(0,24)]

-nombre positif/negatif: [Range(0, int.MaxValue)] / [Range(0, int.MinValue)]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DATATYPE ATTRIBUTE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-date valide: [DataType(DataType.Date)]

-multiligne: [DataType(DataType.MultilineText)]

-type monnaie: [DataType(DataType.Currency)]

-CreditCard : [DataType(DataType.Currency)]

-DateTime : [DataType(DataType.DateTime)]

-EmailAddress : [DataType(DataType.EmailAddress,ErrorMessage ="aaa")]

-ImageUrl : [DataType(DataType.ImageUrl)]

-Password : [DataType(DataType.Password)]

-PhoneNumber : [DataType(DataType.PhoneNumber)]

-PostalCode : [DataType(DataType.PostalCode)]

-Text : [DataType(DataType.Text)]

-Time : [DataType(DataType.Time)]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*COMPARE ATTRIBUTE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[Compare("Email", ErrorMessage= "Email Not Matched")]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*longueur Fixe\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[StringLength(8)]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*longueur maximale \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

longueur maximale de la valeur de données autorisée pour une propriété : [MaxLength(50)]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Phone EmailAddress\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[Phone(ErrorMessage ="phone number")]

[EmailAddress(ErrorMessage = "une address email invalid .")]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*colonne\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-affichage du nom de la colonne dans l’ interface utilisateur :

[Display(Name = “Student Email“)]

public string Email { get; set;}

[Display(Name = "Date of Birth")]

-----------------------------------heritage-------------------------------------------

public class Staff :Passenger

-----------------------------------relation-------------------------------------------

Many to Many \*->\*: public virtual ICollection<Postulant> Postulants { get; set; }

One to Many (Many to One) \* -> 1 : public virtual Entreprise Entreprise { get; set; }

==> //[ForeignKey("PlaneId")] ou bien

public virtual Plane? MyPlane { get; set; }

[ForeignKey("MyPlane")]

public int? PlaneId { get; set; } //prop

//tp5 Q6

//public IList<Passenger> passengers { get; set; }

public virtual IList<Reservation> Reservations { get; set; }

-----------------------------------Config-------------------------------------------

changer le nom de la table : builder.ToTable("MyFlight");

changer le nom de la attribue :

builder.Property(p => p.Capacity).HasColumnName("PlaneCapacity");

builder.HasKey(p => p.PlaneId);

----declarer une relation----

builder

.HasOne(f => f.MyPlane) // ta3 class 1 thot feha one or many w attribue 1

.WithMany(p => p.Flights) // ta3 class 2 thot feha one or many w attribue 2

.HasForeignKey(f => f.PlaneId)

.OnDelete(DeleteBehavior.SetNull);

OR

un simple [ForeignKey(“PlaneId”)] fou9 declamation du var plane

---------------cas Many to Many mnghir table fil west--------------------

//builder

// .HasMany(f => f.passengers)

// .WithMany(p => p.Flights)

// .UsingEntity(ass => ass.ToTable("FP"));

---------cas Many to Many w fama table ili fil west-------------------

//tasna3ha wahdha l classe ili fil west mba3ed tzidha

----------------

public virtual Passenger MyPassenger { get; set; }

public string PassengerId { get; set; }

public virtual Flight MyFlight { get; set; }

public int FlightId { get; set; }

-----------------

builder.HasOne(r => r.MyPassenger)

.WithMany(p => p.Reservations)

.HasForeignKey(r=>r.PassengerId)

.OnDelete(DeleteBehavior.Cascade);

builder.HasOne(r=>r.MyFlight)

.WithMany(f=>f.Reservations)

.HasForeignKey(r=>r.FlightId)

.OnDelete(DeleteBehavior.Cascade);

builder.HasKey(r => new { r.FlightId, r.PassengerId });

=> tansech t3aytelha fil OnModelCreating

--------------- Question mchag3eb TPH (Table Par Hiérarchie).

Configurer l’héritage schématisé dans le diagramme de classe de façon à ce que les

entités seront mappées à

IsTraveller :

=> fil FlightConfig

Exp 1

modelBuilder.Entiy<Membre>()

.HasDiscriminator<string>("Type")

.HasValue<Entraineur>("e")

.HasValue<Joueur>("j")

.HasValue<Membre>("m");

Exp 2

/\*builder.HasDiscriminator<int>("isTraveler")

.HasValue<Passenger>(0)

.HasValue<Traveller>(1)

.HasValue<Staff>(2);\*/

--------------- Question mchag3eb TPT (Table Par Table)

Configurer l'héritage schéématisé dans le diagramme de classe de façonn à ce que les entités

seront mappées sur 3 table :

=> fil AMContext

exp 1

modelBuilder.Entity<Biological>().ToTable("Biologicals");

modelBuilder.Entity<Chemical>().ToTable("Chemicals");

exp 2

//modelBuilder.Entity<Staff>().ToTable("Staffs");

//modelBuilder.Entity<Traveller>().ToTable("Travellers");

---------------------------------LazyLoading-------------------------------------------

optionsBuilder.UseLazyLoadingProxies(); tzidha fil OnConfiguring fil AMContext

ay propriétés de navigation tzidha 9dem public virtual (ay haja de type classe o5ra)

---------------------------------AMContexe-------------------------------------------

public class AMContext : DbContext //herite de DbContext

{

public DbSet<Flight> Flights { get; set; } // pour chaque classe

//parametrer l'acces a la base de donne

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(@"Data Source=(localdb)\mssqllocaldb;

Initial Catalog = Airport;

Integrated Security = true");

//Tp5 Q13

optionsBuilder.UseLazyLoadingProxies(); // LazyLoading

}

//pour appliquer le modification du Flightconfig

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.ApplyConfiguration(new ReservationConfig());

}

//si fama tabdila ala nes lkol

protected override void ConfigureConventions(ModelConfigurationBuilder configurationBuilder)

{

configurationBuilder.Properties<DateTime>()

.HaveColumnType("date");

}

}

---------------------------------Services---------------------------------------------

public class FlightService :Service<Flight>, IFlightService

{}

public IList<Flight> Flights { get; set; } //prop

public IList<DateTime> GetFlightDates(string destination) // linqIntegre

{

//return (from f in Flights

// where f.Destination == destination

// select f.FlightDate).ToList();

return Flights.Where(f => f.Destination == destination) //Methoded'extention

.Select(f => f.FlightDate).ToList();

}

--------------------------Le langage LINQ-------------------------------

var PersoQuery =

from perso in personnes

where ( perso.Budget > 500)

orderby perso.Nom ascending

select new { Name = perso.Nom, Number = perso.Num};

oFirst(): Renvoie le premier élément d'une collection ou le premier élément qui

satisfait une condition.

o FirstOrDefault(): Renvoie le premier élément d'une collection ou le premier

élément qui satisfait une condition. Renvoie une valeur par défaut si l'index est

hors plage.

o Single(): Renvoie le seul élément d'une collection ou le seul élément qui satisfait à

une condition.

o SingleOrDefault(): Identique à Single, sauf qu'il renvoie une valeur par défaut

d'un type générique spécifié, au lieu de lever une exception

Last(): Renvoie le dernier élément d'une collection ou le dernier élément qui

satisfait à une condition.

o LastOrDefault(): Renvoie le dernier élément d'une collection ou le dernier

élément qui satisfait à une condition. Renvoie une valeur par défaut si aucun

élément de ce type n'existe.

o Skip(): Pour indiquer le nombre d’ élément `a ignorer de la sélection.

o Take(): Pour indiquer le nombre d’ élément `a garder de la sélection.

o Average(): calcule la moyenne des éléments numériques de la collection.

o Sum() : calcule la somme des éléments numériques de la collection.

o Max(): renvoie le plus grand élément numérique d'une collection.

o Count: renvoie le nombre d'éléments de la collection ou le nombre d'éléments

qui ont satisfait à la condition donnée.

-------------------------Les méthodes anonymes-------------------------

public void ShowFlightDetails(Plane plane)

{

var result = from f in Flights

where f.MyPlane.PlaneId == plane.PlaneId

select new { f.Destination, f.FlightDate }; //type anonyme

foreach (var item in result)

{

Console.WriteLine("destination : " + item.Destination +

"date :" + item.FlightDate);

}

}

---------------------------------------------------------------------------

internal class ReservationConfig : IEntityTypeConfiguration<Reservation>

{

public void Configure(EntityTypeBuilder<Reservation> builder)

{

builder.HasOne(r => r.MyPassenger)

.WithMany(p => p.Reservations)

.HasForeignKey(r=>r.PassengerId)

.OnDelete(DeleteBehavior.Cascade);

builder.HasOne(r=>r.MyFlight)

.WithMany(f=>f.Reservations)

.HasForeignKey(r=>r.FlightId)

.OnDelete(DeleteBehavior.Cascade);

builder.HasKey(r => new { r.FlightId, r.PassengerId });

}

}

---------------------------------------------------------------------------

Add-Migration

Update-Database

--------------------------------Configuration------------------------------------------

internal class PlaneConfig : IEntityTypeConfiguration<Plane>

{

public void Configure(EntityTypeBuilder<Plane> builder)

{

builder.ToTable("MyPlanes"); //modifcation du nom

builder.Property(p => p.Capacity).HasColumnName("PlaneCapacity");

builder.HasKey(p=>p.PlaneId); // fixsage du cle primere

//changer le nom d’une colonne

builder.Property(p => p.Capacity)

.HasColumnName("PlaneCapacity");

}

}

--------------------------------Relation--------------------------------

//builder

// .HasMany(f => f.passengers)

// .WithMany(p => p.Flights)

// .UsingEntity(ass => ass.ToTable("FP"));

builder

.HasOne(f => f.MyPlane)

.WithMany(p => p.Flights)

.HasForeignKey(f => f.PlaneId)

.OnDelete(DeleteBehavior.SetNull);

--------------------------Interface----------------------------

public interface IServiceActivite : IService<Activite>

{

}

----------------------Service--------------------

public class ServiceActivite : Service<Activite>, IServiceActivite

{

public ServiceActivite(IUnitOfWork unitOfWork) : base(unitOfWork)

{

}

}

-----------------View----------------------

//liste roulante :

<div class="form-group">

<label asp-for="ConseillerFk" class="control-label"></label>

<select asp-for="ConseillerFk" class="form-control" asp-items="ViewBag.Conseiller"></select>

<span asp-validation-for="ConseillerFk" class="text-danger"></span>

</div>

////searche bar :

<form asp-action="index">

<fieldset>

<legend> Recherche</legend>

<input type="string" name="login"/>

<input type="submit" value="serach" />

</fieldset>

</form>

/////link eli chyhezna ll details

@Html.ActionLink("Conseiller", "Details", new { id=item.ConseillerFk})

---------------Controller-------------------

//tasna3 instance mel service

IServiceClient serviceClient;

IServiceConseiller serviceConseiller;

//tgenerie el constructeur

public ClientController(IServiceClient serviceClient, IServiceConseiller serviceConseiller)

{

this.serviceClient = serviceClient;

this.serviceConseiller = serviceConseiller;

}

///fonction index wa9t 3andi searche

public ActionResult Index(string? login)

{

if (login == null)

return View(serviceClient.GetAll());

else

return View(serviceClient.GetMany(cl =>cl.Login==login));

}

// fel create wa9t 3andi liste roulante :

//Partie 1

// GET: ClientController/Create

public ActionResult Create()

{

ViewBag.Conseiller = new SelectList(serviceConseiller.GetAll(), "ConseillerId", "Nom");

return View();

}

//partie 2

// POST: ClientController/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Client client)

{

try

{

serviceClient.Add(client);

serviceClient.Commit();

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

-----------------exemple de service---------

public double DurationAverage(string destination)

{

var flightsToDestination = Flights.Where(flight => flight.Destination.Equals(destination, StringComparison.OrdinalIgnoreCase)).ToList();

if (flightsToDestination.Count == 0)

{

return 0;

}

double averageDuration = flightsToDestination.Average(flight => flight.EstimatedDuration);

return averageDuration;

}

////////////

public IList<Prestataire> GetBestPrestataire()

{

return (from p in Prestataires

where p.zone == Zone.Raoued

orderby p.Note descending

select p).Take(3).ToList();

}

//////////

public double GetBestPrestataire()

{

return (from p in Prestations

where p.PrestationType == Type.Coiffure

select p.Prix).Average();

}

/////

public List<Traveller> SeniorTravellers(Flight flight)

{

var travellers = flight.Tickets.OfType<Traveller>();

var seniorTravellers = travellers

.OrderBy(traveller => traveller.BirthDate)

.Take(3)

.ToList();

return seniorTravellers;

}

//////

public int ProgrammedFlightNumber(DateTime startDate)

{

DateTime endDate = startDate.AddDays(7);

int flightCount = Flights.Count(flight => flight.FlightDate >= startDate && flight.FlightDate < endDate);

return flightCount;

}

-----------AMContexte--------------

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

//modelBuilder.ApplyConfiguration(new FlightConfiguration ());

//cle primaire composer dans la classe Reservation

modelBuilder.Entity<Reservation>().HasKey(r=>new { r.ClientFk , r.PackFk});

//Relation 1 to Many

modelBuilder.Entity<Client>().HasOne(c=>c.Conseiller).WithMany(cl=> cl.Clients).HasForeignKey(c=>c.ConseillerFk).OnDelete(DeleteBehavior.Cascade);

//Configurer l’héritage schématisé dans le diagramme de façon à ce que les trois entités Flower, NaturalFlower et ArtificialFlower soient mappées dans trois tables différentes.

modelBuilder.Entity<ArtificialFlower>().ToTable("ArtificialFlowers");

modelBuilder.Entity<NaturalFlower>().ToTable("naturalFlowers");

}

public IList<IGrouping<Secteur, Entreprise>> ParSecteur(string ville)

{

return GetAll().Where(e => e.Adresse.Ville== ville).GroupBy(e => e.Secteur).ToList();

}

public int PME()

{

return GetAll().Where(e => (e.Effectif >= 10 && e.Effectif <= 250 && e.ChiffreAffaire < 50000) ).Count();

}

public IList<Entreprise> Top2(int typeC)

{

return GetAll().Where(e => e.Secteur == (Secteur)typeC).OrderBy(e => e.Offres.Count).Take(2).ToList();

}

------------------------------------------------------------------------

AMContext : OnModelCreating

// Q : convention personnalisée qui configure toutes les propriétés dont le nom commence

// par "Mail" comme étant obligatoires

// Parcourir toutes les entités dans le modèle

foreach (var entityType in modelBuilder.Model.GetEntityTypes())

{

// Parcourir toutes les propriétés de l'entité

foreach (var property in entityType.GetProperties())

{

// Vérifier si la propriété est de type string et commence par "Mail"

if (property.ClrType == typeof(string) && property.Name.StartsWith("Mail"))

{

property.IsNullable = false; // Rend la propriété obligatoire

}

}

}

// Populate the dropdown with enum values

ViewBag.TypeList = Enum.GetValues(typeof(ApplicationCore.Domain.Type))

.Cast<ApplicationCore.Domain.Type>()

.Select(t => new SelectListItem

{

Value = t.ToString(),

Text = t.ToString()

})

.ToList();

------------------delete------------------

// GET: ElecteurController/Delete/5

public ActionResult Delete(int id)

{

return View(serviceElecteur.GetById(id));

}

// POST: ElecteurController/Delete/5

[HttpPost] [ValidateAntiForgeryToken]

public ActionResult Delete(int id, IFormCollection collection)

{

try {

serviceElecteur.Delete(serviceElecteur.GetById(id)); serviceElecteur.Commit();

return RedirectToAction(nameof(Index));

}

catch { return View(); } } }