Revv

```
 [DataType(DataType.DateTime)] // La propriété DateReservation d

[Range(1,4)]// La propriété NbPersonnes doit appartenir à l’inte

[Key] // La propriété Identifiant doit être la clé primaire de l

 [Required(ErrorMessage ="Login is required")] //La propriété Lo
 être obligatoire avec affichage d’un message d’erreur si elle
n’est pas saisie par l’utilisateur.

[Required]
[DataType(DataType.Password)]

La propriété Password doit être obligatoire et la valeur saisie
l’utilisateur doit êtrecachée.

        [ForeignKey("Pack")]


[Display(Name = "Le motif de l'admission")]
[DataType(DataType.MultilineText)]
La propriété MotifAdmission doit être Multiligne et affiché «Le
[Key]
[StringLength(7)]  un champ de 7 caractères
   [MinLength(3, ErrorMessage = "Longeur Minimale 3!")]
   [MaxLength(25, ErrorMessage = "Longeur Maximale 25!")]

   [Display(Name ="Date of Birth")]  affichée “Date of Birth”

   [DataType(DataType.EmailAddress)]
```

```
    [RegularExpression("^[0-9]{8}$")]


     [Range(0, int.MaxValue)]  :  un entier positif
     [DataType(DataType.Currency)] : ■ une valeur monétaire


     [Required(ErrorMessage = "Le nom est requis")]
    [Column("ProductName")] // mappe vers des column
```

```
 Add-migration mig1
 update-database
```

```
■ PlaneId est la clé primaire de la table
■ Le nom de la table correspondante à l’entité Plane dans la bas
“MyPlanes”
■ Le nom de la colonne correspondante à la propriété Capacity da
données doit être PlaneCapacity

  public void Configure(EntityTypeBuilder<Plane> builder)
  {
      builder.HasKey(p => p.PlaneId);
      builder.ToTable("myPlanes");
      builder.Property(p => p.Capacity).HasColumnName("PlaneCapa

  }
```

```
 Ajouter la class PassengerConfiguration dans le dossier Configu
■ Configurer le type d’entité détenu FullName
■ La propriété FirstName a une longueur maximale de 30 et le nom
correspondante à cette propriété dans la base de données doit êt

  builder.OwnsOne(p => p.FullName,
```

```
       full=> {
           full.Property(f=>  f.FirstName).HasMaxLength(30).HasC
           });

  // faire date par defaut in OnModelCreating

     foreach (var entityType in modelBuilder.Model.GetEntityType
    {
        foreach (var property in entityType.GetProperties())
        {
            if (property.ClrType == typeof(DateTime))
            {
                modelBuilder.Entity(entityType.ClrType).Property
            }
        }
    }

    //configurer tous les propeites de types string pour qu'lles
    ConfigureConventions:

configurationBuilder.Properties<string>().HaveMaxLength(200);


//Pour configurer la colonne Discriminator de la hiérarchie d'he
entre les classes Produit, Chimique et Biologique afin qu'elle s
 de type char et qu'elle prenne la valeur 'C'
  si le type de Produit est Chimique, 'B' si le type de Produit
  est Biologique et 'P' sinon

protected override void OnModelCreating(ModelBuilder modelBuilde
{
    modelBuilder.Entity<Produit>()
        .HasDiscriminator<char>("TypeProduit")
        .HasValue<Produit>('P')
        .HasValue<Chimique>('C')
        .HasValue<Biologique>('B');
```

```
    base.OnModelCreating(modelBuilder);
}
```

```
[Display(Name = "Nom")] : Spécifie le nom à afficher pour une pr
[Required] : Indique qu'une propriété est obligatoire (non nulla
[StringLength(maximumLength, MinimumLength = x)] : Spécifie la l
[Range(minimum, maximum)] : Spécifie une plage de valeurs valide
[RegularExpression(pattern)] : Spécifie une expression régulière
[DataType(DataType.Date)] : Indique le type de données d'une pro
[EmailAddress] : Indique que la valeur d'une propriété doit être
[Url] : Indique que la valeur d'une propriété doit être une URL
[Compare("otherProperty")] : Compare la valeur d'une propriété a
[MaxLength(length)] : Spécifie la longueur maximale d'une chaîne
[MinLength(length)] : Spécifie la longueur minimale d'une chaîne
[DisplayFormat(DataFormatString = "{0:format}")] : Spécifie le f
[Key] : Indique que la propriété est une clé primaire dans une c
[ForeignKey("propertyName")] : Spécifie le nom de la propriété c
[NotMapped] : Indique que la propriété ne doit pas être mappée à
[Editable(false)] : Indique que la propriété ne peut pas être mo
[DisplayFormat(NullDisplayText = "Texte")] : Spécifie le texte à
[DataType(DataType.MultilineText)] : Indique que la propriété es
[DisplayFormat(DataFormatString = "{0:C}")] : Spécifie le format
[Timestamp] : Indique que la propriété est utilisée pour gérer l
```

```
using System;
using System.ComponentModel.DataAnnotations;

public class Person
{
    public int Id { get; set; }

    [Display(Name = "Nom complet")]
    [Required(ErrorMessage = "Le nom est requis")]
```

```
    public string FullName { get; set; }

    [StringLength(50, MinimumLength = 2, ErrorMessage = "Le prén
    public string FirstName { get; set; }

    [Range(0, 120, ErrorMessage = "L'âge doit être compris entre
    public int Age { get; set; }

    [RegularExpression(@"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z
    public string Email { get; set; }

    [DataType(DataType.Date)]
    public DateTime BirthDate { get; set; }

    [Url(ErrorMessage = "URL non valide")]
    public string Website { get; set; }

    [Compare("PasswordConfirmation", ErrorMessage = "Les mots de
    public string Password { get; set; }

    [DisplayFormat(DataFormatString = "{0:C}")]
    public decimal Salary { get; set; }

    [NotMapped]
    public string PasswordConfirmation { get; set; }

    [Editable(false)]
    public string SecretNotes { get; set; }

    [DisplayFormat(NullDisplayText = "Non spécifié")]
    public string Bio { get; set; }

    [DataType(DataType.MultilineText)]
    public string Comments { get; set; }

    [Timestamp]
```

```
    public byte[] RowVersion { get; set; }
}
```

```
public class Product
{
    public int Id { get; set; }

    [ScaffoldColumn(false)] //ne doit pas être incluse dans les
    public string InternalCode { get; set; }

    [Display(Name = "Nom du produit")]
    [Required(ErrorMessage = "Le nom est requis")]
    public string Name { get; set; }

    [StringLength(100, ErrorMessage = "La description ne doit pa
    public string Description { get; set; }

    [Range(0, 1000, ErrorMessage = "Le prix doit être compris en
    [DataType(DataType.Currency)]
    public decimal Price { get; set; }

    [RegularExpression(@"^[A-Z]{3}-[0-9]{3}$", ErrorMessage = "I
    public string ProductCode { get; set; }

    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFor
    public DateTime ReleaseDate { get; set; }

    [UIHint("MultilineText")]
    public string Comments { get; set; }

    [BindNever] //Indique que la propriété ne doit pas être incl
    public string HiddenProperty { get; set; }
```

```
    [ReadOnly(true)]
    public string ReadOnlyProperty { get; set; }

    [ConcurrencyCheck] //Indique que la propriété est utilisée p
    public int Version { get; set; }

    [InverseProperty("Products")] //Spécifie la propriété invers
    public virtual Category Category { get; set; }

    [DatabaseGenerated(DatabaseGeneratedOption.Identity)] //Indi
    public int AutoGeneratedId { get; set; }

    [Table("CustomProducts")] // Spécifie le nom de la table ass
    public class CustomProduct { }
}
```

```
. toutes les propriétés de type DateTime pour qu’elles soient ma
colonnes de type Date. (0.75 pt)

configurationBuilder.Properties<DateTime>().HaveColumnType("date
```