ASP.NET MVC : modèle & scaffolding

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

Le modèle

Le modèle

- constitue le troisième composant d'un modèle MVC
- est composé de deux parties : une partie structure de données et une seconde gestion de données

Le modèle

Le modèle

- constitue le troisième composant d'un modèle MVC
- est composé de deux parties : une partie structure de données et une seconde gestion de données

Un ORM pour le modèle

- permet de construire la partie structure de données (entités)
- permet aussi la gestion des données (via un gestionnaire d'entité)

Le modèle

Entity Framework (EF)

- est le premier framework pour les applications .NET
- peut être installé via le gestionnaire de packages NuGet

Le modèle

Entity Framework (EF)

- est le premier framework pour les applications .NET
- peut être installé via le gestionnaire de packages NuGet

Trois approches possibles

- Code First
- Database First
- Model First

Le modèle

Avec ASP.NET MVC 5 (et EF)

- On prépare les entités et le contexte
- Il nous génère les vues, les contrôleurs...

Le modèle

Étapes (Code First)

- Créer un nouveau projet
- Installer EF
- Préparer les entités
- Générer le contexte
- Créer le contrôleur et les vues associées qui assureront le CRUD.

Le modèle

Créons une première entité `Personne` dans le `Models`

```
public class Personne
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }
    public string Nom { get; set; }
    public string Prenom { get; set; }
    public int Age { get; set; }
    public virtual ICollection<Adresse> Adresses { get; set; }
    public Personne()
    {
        Adresses = new List<Adresse>();
    }
}
```

Le modèle

Créons une première entité `Personne` dans le `Models`

```
public class Personne
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }
    public string Nom { get; set; }
    public string Prenom { get; set; }
    public int Age { get; set; }
    public virtual ICollection<Adresse> Adresses { get; set; }
    public Personne()
    {
        Adresses = new List<Adresse>();
    }
}
```

Pour les décorateurs de validation, il faut utiliser l'espace de noms suivant :

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

Le modèle

Créons une deuxième entité `Adresse` dans le répertoire `Models`

```
public class Adresse
{
    [Key]
    [Column(Order=1)]
    public int Rue { get; set; }

    [Key]
    [Column(Order=2)]
    public string Ville{ get; set; }

    [Key]
    [Column(Order=3)]
    public string CodeP{ get; set; }

    [ForeignKey("Personne")]
    public int PersonneId { get; set; }
    public Personne Personne { get; set; }
}
```

Le modèle

N'oublions pas de

- Installer EF
- Préparer le contexte (clic droit sur le projet et aller `Ajouter > Nouvel élément > Données > ADO.NET Entity Data Model > Modèle vide Code First`)

Le modèle

Mettons à jour le contexte (la classe `Model1`)

```
using CoursModelEf.Models;

public class Model1 : DbContext
{
    public Model1()
        : base("name=Model1")
    {
    }
    public virtual DbSet<Personne> Personne { get;
       set; }
    public virtual DbSet<Adresse> Adresse { get; set
       ; }
}
```

Le modèle

Pour générer le CRUD

- Faire clic droit sur le nom du projet dans l'`Explorateur de solutions` et choisir `Générer`
- Faire clic droit sur le répertoire `Controllers`
- Aller dans `Ajouter > Contrôleur`
- Choisir `MVC 5 Controller with view using Entity Framework`
- Dans `Model class :` choisir `Adresse` ensuite choisir le contexte (`Model1`)
- Valider

Le modèle

Pour générer le CRUD

- Faire clic droit sur le nom du projet dans l'`Explorateur de solutions` et choisir `Générer`
- Faire clic droit sur le répertoire `Controllers`
- Aller dans `Ajouter > Contrôleur`
- Choisir `MVC 5 Controller with view using Entity Framework`
- Dans `Model class :` choisir `Adresse` ensuite choisir le contexte (`Model1`)
- Valider

Refaire la même chose pour `Personne` mais en cochant toutes les cases de `Views` avant de valider

Le modèle

Résultat : tout a été généré

- Toutes les actions dans les deux contrôleurs
- Toutes les vues aussi (Avec Bootstrap pour les vues appelées par `PersonneController` car nous avons cochés les trois cases)
- Les liens de navigations...

Le modèle

Résultat : tout a été généré

- Toutes les actions dans les deux contrôleurs
- Toutes les vues aussi (Avec Bootstrap pour les vues appelées par `PersonneController` car nous avons cochés les trois cases)
- Les liens de navigations...

Attention, certaines actions et vues sont à corriger pour `Adresse` à cause de la clé primaire composée

Le modèle

Pour mettre à jour le menu, dans `_Layout.cshtml` remplacer

```
@Html.ActionLink("Application name", "Index", "Home"
    , new { area = "" }, new { @class = "navbar-brand
    " })
```

par

```
@Html.ActionLink("Carnet d'adresse", "Index", "Home"
    , new { area = "" }, new { @class = "navbar-brand
    " })
@Html.ActionLink("Personne", "Index", "Personnes",
    new { area = "" }, new { @class = "navbar-brand"
    })
@Html.ActionLink("Adresse", "Index", "Adresses", new
     { area = "" }, new { @class = "navbar-brand" })
```