

# Modèle en ASP.NET MVC

---

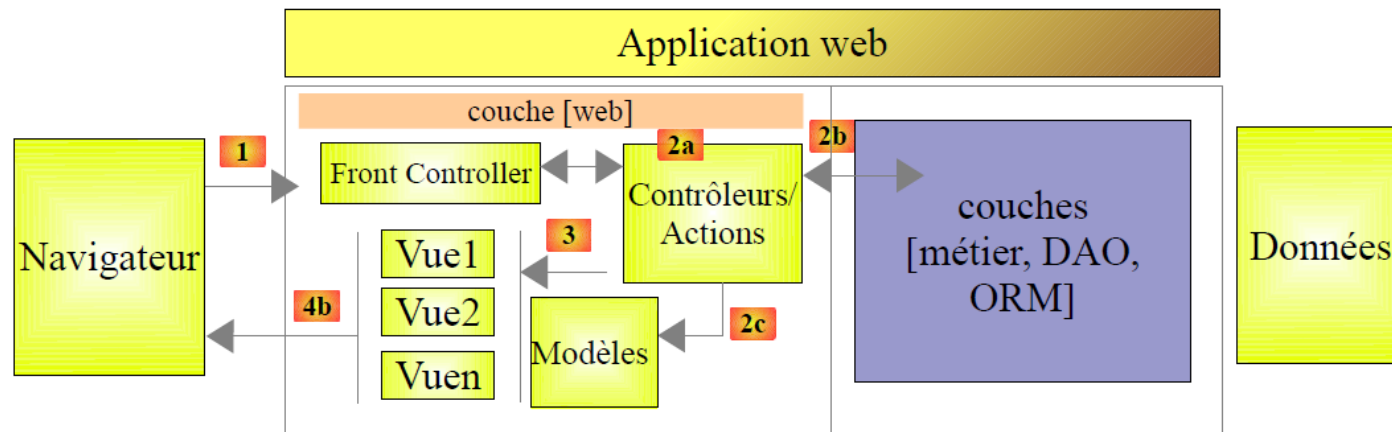
MOHAMED AIROUCHE

# Le modèle

➤ **Le modèle** est une collection de classes qui modélisent les données utilisées dans une application Web.

Précisons le lien entre l'architecture Web MVC et l'architecture en couches. Selon la définition qu'on donne au modèle, ces deux concepts sont liés ou non.

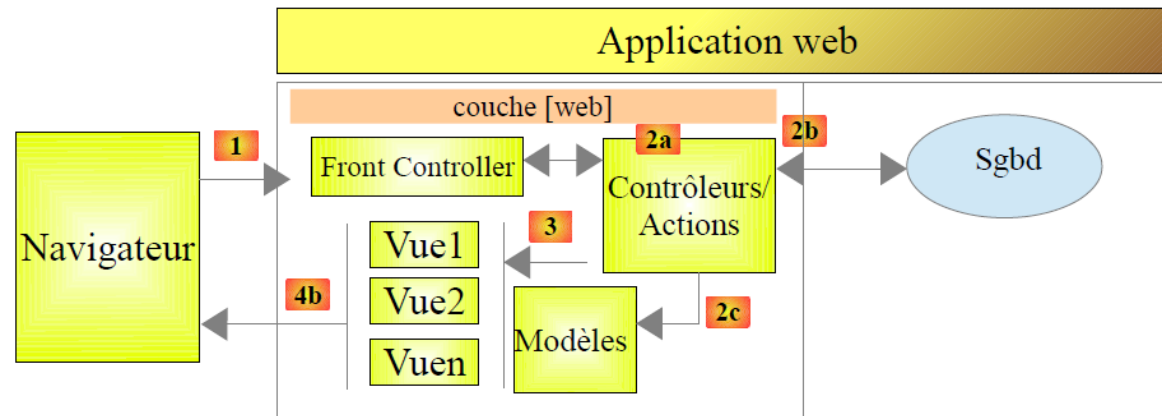
ASP.NET MVC implémente le modèle d'architecture MVC (Modèle – Vue – Contrôleur) de la façon suivante :



<http://sergetahe.com/cours-tutoriels-de-programmation/wp-content/uploads/documents/Introduction%C3%A0%20ASP.NET%20MVC%20par%20l'exemple%20-%202013.pdf>

# Le modèle

Dans une application Web ASP.NET MVC à une couche, la couche [Web] s'occupera de tout : présentation, métier, accès aux données. Ce sont les actions qui feront ce travail.

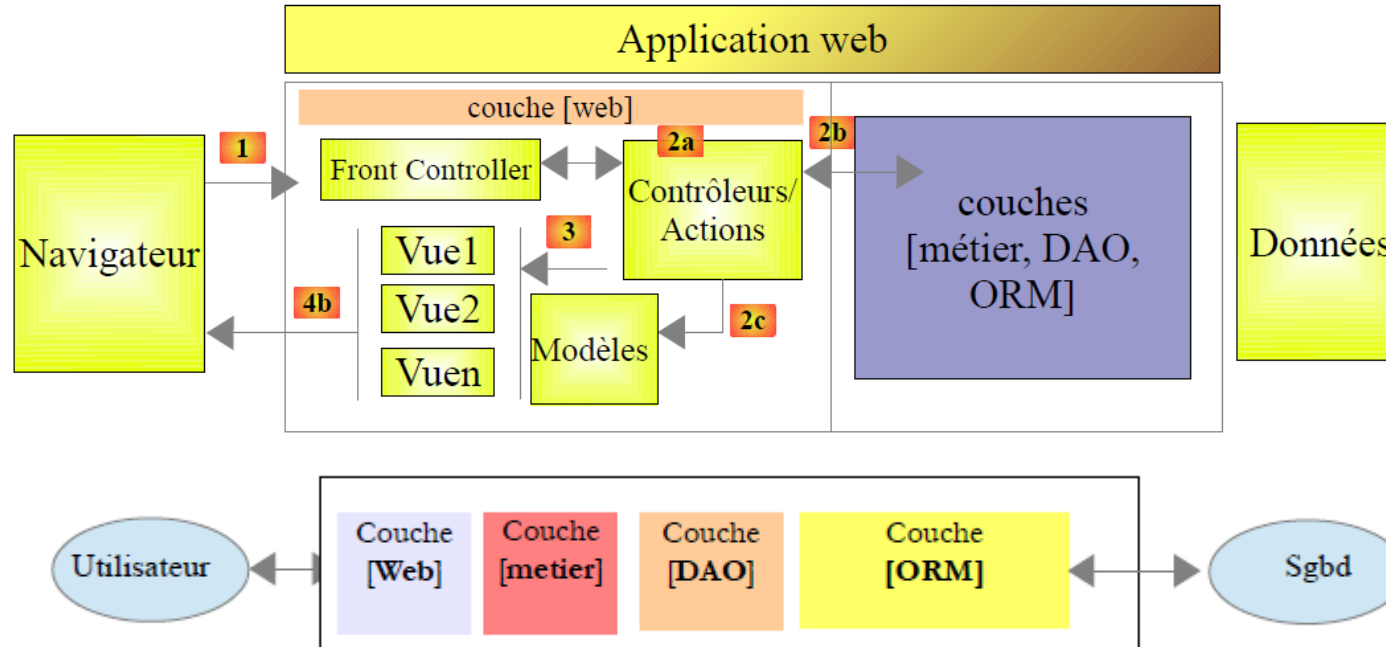


- **Le modèle** représente les données spécifiques au domaine et la logique métier dans l'architecture MVC.

<http://sergetahe.com/cours-tutoriels-de-programmation/wp-content/uploads/documents/Introduction%20C3%A0%20ASP.NET%20MVC%20par%20l'exemple%20-%202013.pdf>

# Le modèle

Dans une application Web ASP.NET MVC avec une architecture Web multicouche :



La couche [Web] peut être implémentée avec ASP.NET MVC et on a alors une architecture en couches avec une couche [Web] de type MVC.

➤ **Le modèle** est l'ensemble des données affichées par la vue.

<http://sergetahe.com/cours-tutoriels-de-programmation/wp-content/uploads/documents/Introduction%20%C3%A0%20ASP.NET%20MVC%20par%20l'exemple%20-%202013.pdf>

# Le modèle

---

Beaucoup d'auteurs considèrent que ce qui est à droite de la couche [Web] forme le modèle M du MVC.

Pour éviter les ambiguïtés on peut parler :

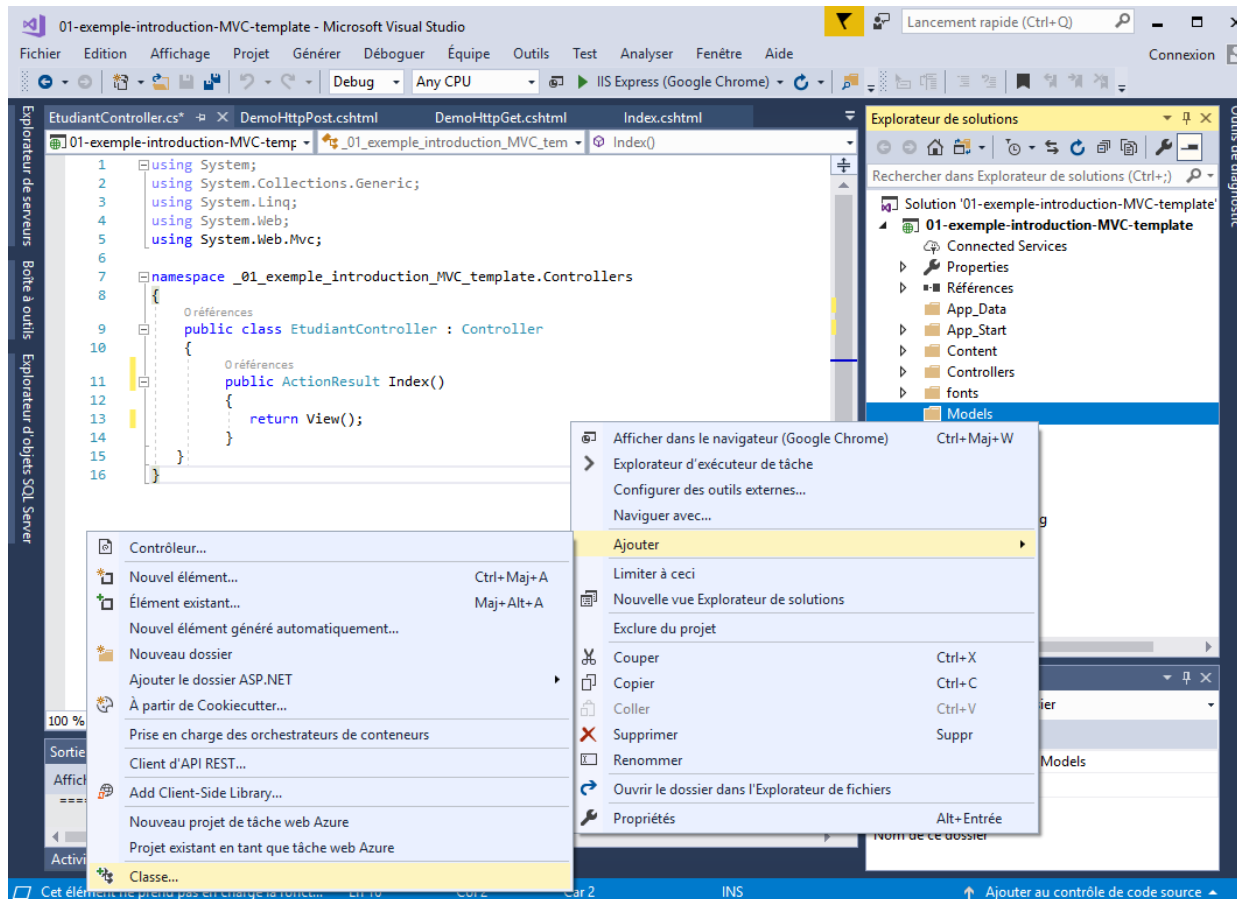
- du **modèle du domaine** ou **modèle de l'application** lorsqu'on désigne tout ce qui est à droite de la couche [Web].
- du **modèle de la vue** lorsqu'on désigne les données affichées par une vue V.

Dans un premier temps, le terme **modèle** désignera exclusivement le **modèle d'une vue**. Par la suite, **Le modèle** est une collection de classes qui modélisent les données utilisées dans une application Web.

<http://sergetahe.com/cours-tutoriels-de-programmation/wp-content/uploads/documents/Introduction%20%C3%A0%20ASP.NET%20MVC%20par%20l'exemple%20-%202013.pdf>

# Le modèle - Exemple

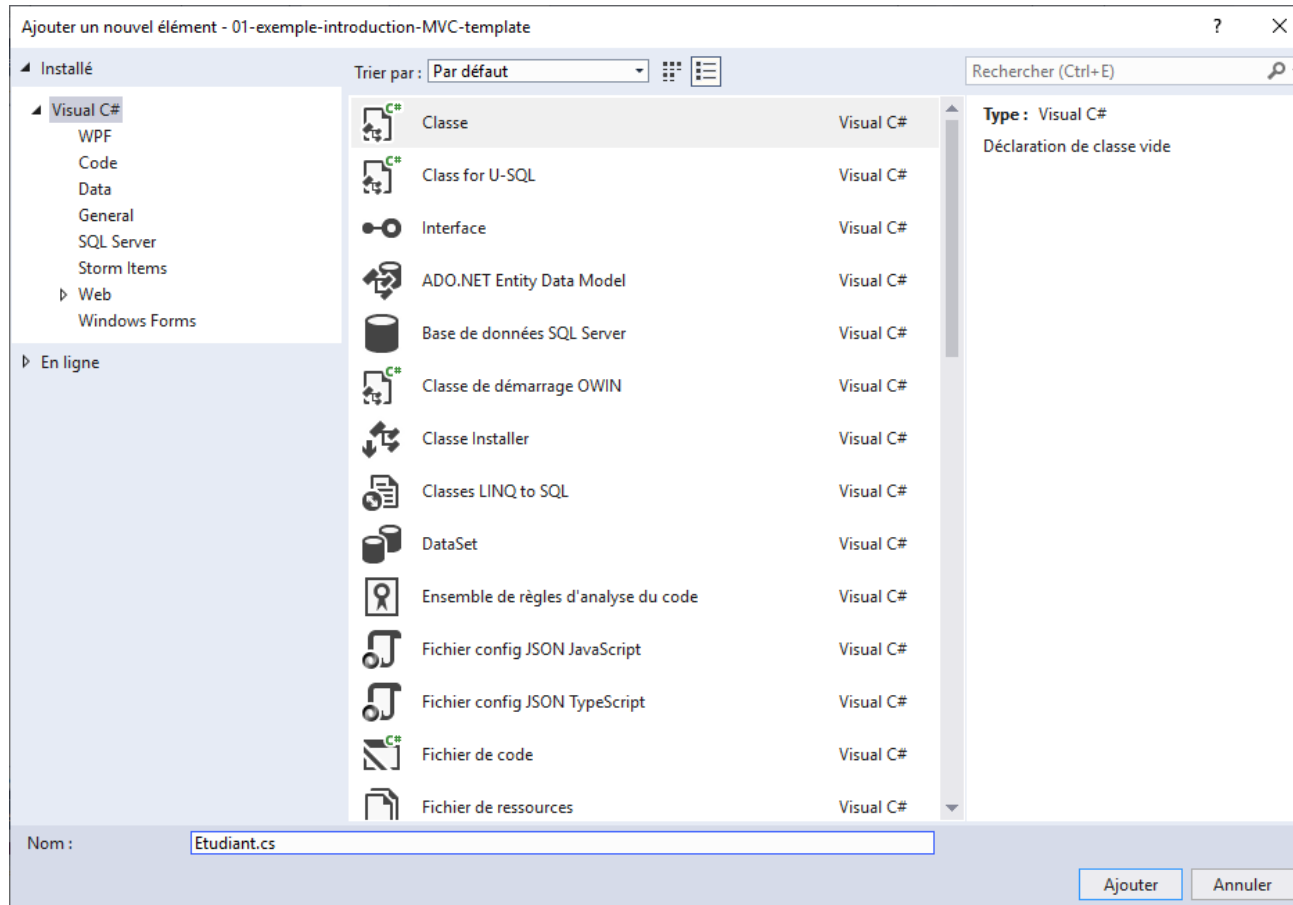
- Pour Ajouter un modèle, dans l'**Explorateur de solutions**, cliquez avec le bouton droit sur le dossier *Models*, sélectionnez **Ajouter**, puis **classe**.



<https://docs.microsoft.com/fr-ca/aspnet/mvc/overview/getting-started/introduction/adding-a-model>

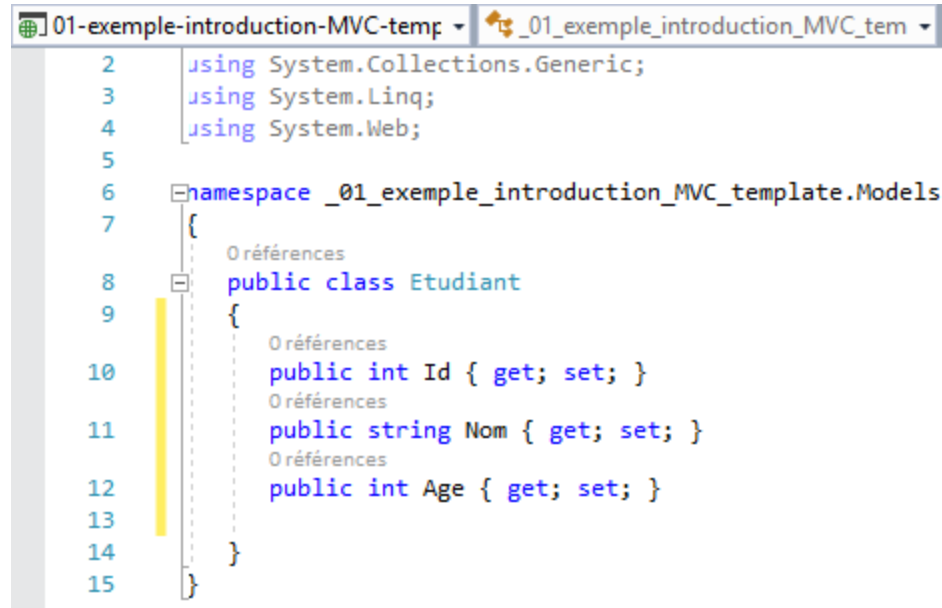
# Le modèle - Exemple

- Ajout d'un modèle Etudiant dans le dossier *Models* (entrez le nom de la classe "Etudiant").



# Le modèle - Exemple

- Ajoutez les trois propriétés suivantes à la classe Etudiant.



The screenshot shows a Visual Studio code editor with a file named `01-exemple-introduction-MVC-temp` open. The code is in C# and defines a namespace `_01_exemple_introduction_MVC_template.Models`. Inside this namespace, there is a public class `Etudiant`. The class has three public properties: `Id` (int), `Nom` (string), and `Age` (int). Each property has a getter and a setter. The code is as follows:

```
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5
6 namespace _01_exemple_introduction_MVC_template.Models
7 {
8     public class Etudiant
9     {
10         public int Id { get; set; }
11         public string Nom { get; set; }
12         public int Age { get; set; }
13     }
14 }
15
```



# Passer des données de contrôleur à la vue

---

Dans une application ASP.NET MVC, le contrôleur permet de passer les données d'un modèle à une vue pour qu'elles soit affichées. Ces données peuvent être transféré en utilisant :

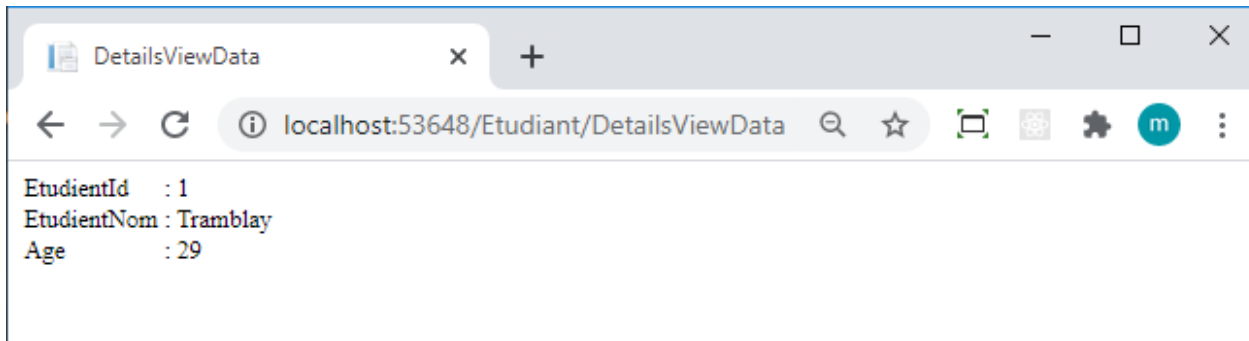
- ViewData
  - ViewBag
  - TempData
- 
- On peut également utiliser le modèle lié à la vue: Les vues fortement typées
- 
- Afin de créer une vue fortement typée dans ASP.NET MVC, nous devons spécifier le type de modèle dans la vue à l'aide de la directive @model.

# Exemple d'utilisation de ViewData

## ➤ L'action de contrôleur

```
//utilisation de ViewData pour transferer les données de l'action vers la vue
Or références
public ActionResult DetailsViewData()
{
    Etudiant etudiant = new Etudiant { Id = 1, Nom = "Tramblay", Age = 29 };
    ViewData["Id"] = etudiant.Id;
    ViewData["Nom"] = etudiant.Nom;
    ViewData["Age"] = etudiant.Age;
    return View();
}
```

## ➤ L'affichage de la vue dans le navigateur



## ➤ La vue

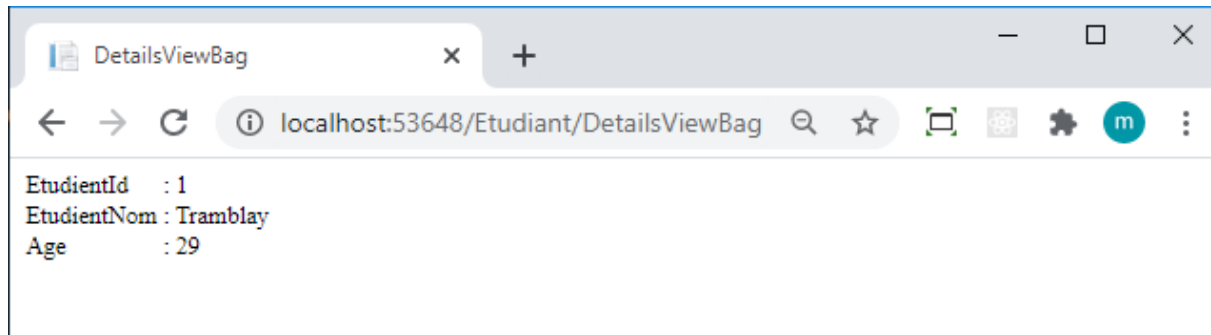
```
8 <html>
9 <head>
10     <meta name="viewport" content="width=device-width" />
11     <title>DetailsViewData</title>
12 </head>
13 <body>
14     <table>
15         <tr>
16             <td>
17                 EtudiantId
18             </td>
19             <td>
20                 : @ViewData["Id"]
21             </td>
22         </tr>
23         <tr>
24             <td>
25                 EtudiantNom
26             </td>
27             <td>
28                 : @ViewData["Nom"]
29             </td>
30         </tr>
31         <tr>
32             <td>
33                 Age
34             </td>
35             <td>
36                 : @ViewData["Age"]
37             </td>
38         </tr>
39     </table>
40 </body>
41 </html>
```

# Exemple d'utilisation de ViewBag

## ➤ L'action de contrôleur

```
//utilisation de ViewBag pour transferer les données de l'action vers la vue
Références
public ActionResult DetailsViewBag()
{
    Etudiant etudiant = new Etudiant { Id = 1, Nom = "Tramblay", Age = 29 };
    ViewBag.etudiant = etudiant;
    return View();
}
```

## ➤ L'affichage de la vue dans le navigateur



## ➤ La vue

```
8 <html>
9 <head>
10     <meta name="viewport" content="width=device-width" />
11     <title>DetailsViewBag</title>
12 </head>
13 <body>
14     <table>
15     <tr>
16         <td>
17             EtudiantId
18         </td>
19         <td>
20             : @ViewBag.etudiant.Id
21         </td>
22     </tr>
23     <tr>
24         <td>
25             EtudiantNom
26         </td>
27         <td>
28             : @ViewBag.etudiant.Nom
29         </td>
30     </tr>
31     <tr>
32         <td>
33             Age
34         </td>
35         <td>
36             : @ViewBag.etudiant.Age
37         </td>
38     </tr>
39 </table>
40 </body>
41 </html>
```

# Les helpers fortement typés

---

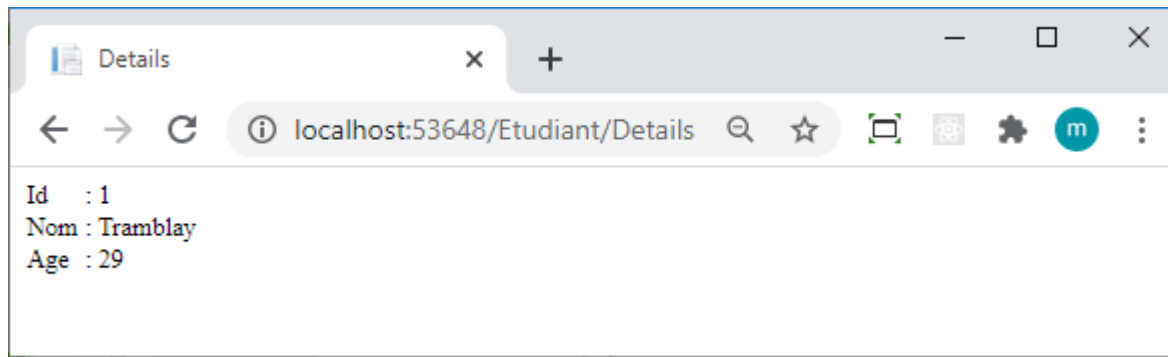
- Le modèle permet d'utiliser les helpers fortement typés
- Les helpers fortement typés fonctionnent comme les helpers standards, mais plutôt que de travailler avec des chaînes de caractères, ils fonctionnent grâce au pouvoir des expressions lambdas.
- En général, il y a un helper fortement typé pour chaque helper disponible, ils sont simplement suffixés par For.
- Exemple:  
`@Html.TextBoxFor(model => model.Nom)`

# Exemple d'utilisation d'une vue fortement typée

## ➤ L'action de contrôleur

```
//utilisation d'une vue fortement typée pour transferer les données de l'action vers la vue
0 références
public ActionResult Details()
{
    Etudiant etudiant = new Etudiant { Id = 1, Nom = "Tramblay", Age = 29 };
    return View(etudiant);
}
```

## ➤ L'affichage de la vue dans le navigateur



## ➤ La vue

```
@model Exemple01MVC.Models.Etudiant
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Details</title>
</head>
<body>
    <table>
        <tr>
            <td>
                @Html.DisplayNameFor(model => model.Id)
            </td>
            <td>
                : @Html.DisplayFor(model => model.Id)
            </td>
        </tr>
        <tr>
            <td>
                @Html.DisplayNameFor(model => model.Nom)
            </td>
            <td>
                : @Html.DisplayFor(model => model.Nom)
            </td>
        </tr>
        <tr>
            <td>
                @Html.DisplayNameFor(model => model.Age)
            </td>
            <td>
                : @Html.DisplayFor(model => model.Age)
            </td>
        </tr>
    </table>
</body>
</html>
```