

Support de cours

**Architecture des SI I**

Chapitre 6 : DOTNET MVC WEB

Table des matières

[**Contrôleur de vue de modèle (MVC)** 2](#_Toc113367573)

[MVC avec ASP.NET 2](#_Toc113367574)

[**Qu'est - ce Razor ?** 3](#_Toc113367575)

[Razor Syntaxe 3](#_Toc113367576)

[4](#_Toc113367577)

[ASP.NET Razor - C # Variables 4](#_Toc113367578)

[ASP.NET Razor - C # boucles et tableaux 8](#_Toc113367579)

[ASP.NET Core Tag Helpers in Forms: 10](#_Toc113367580)

[Cache Tag Helper 14](#_Toc113367581)

[liste déroulante dynamique 15](#_Toc113367582)

[liste déroulante statique 17](#_Toc113367583)

[Les vues partielles 18](#_Toc113367584)

# Contrôleur de vue de modèle (MVC)

MVC est un modèle de conception utilisé pour découpler l'interface utilisateur (vue), les données (modèle) et la logique d'application (contrôleur). Ce modèle aide à réaliser la séparation des préoccupations.

En utilisant le modèle MVC pour les sites Web, les demandes sont acheminées vers un contrôleur chargé de travailler avec le modèle pour effectuer des actions et/ou récupérer des données. Le contrôleur choisit la vue à afficher et lui fournit le modèle. La vue affiche la page finale, en fonction des données du modèle.

# MVC avec ASP.NET

ASP.NET vous offre un moyen puissant, basé sur des modèles, de créer des sites Web dynamiques à l'aide du modèle MVC qui permet une séparation nette des préoccupations.

#### Responsabilité du modèle

Le modèle d’une application MVC représente l’état de l’application, ainsi que la logique métier ou les opérations à effectuer. La logique métier doit être encapsulée dans le modèle, ainsi que toute autre logique d’implémentation pour la persistance de l’état de l’application. En général, les vues fortement typées utilisent des types ViewModel conçus pour contenir les données à afficher sur cette vue. Le contrôleur crée et remplit ces instances de ViewModel à partir du modèle.

#### Responsabilité de la vue

Les vues sont responsables de la présentation du contenu via l’interface utilisateur. Ils utilisent le moteur d’affichage pour incorporer du [Razor](https://docs.microsoft.com/fr-fr/aspnet/core/mvc/overview?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0#razor-view-engine) code .NET dans le balisage HTML. Il doit exister une logique minimale dans les vues, et cette logique doit être liée à la présentation du contenu. Si vous avez besoin d’exécuter une grande partie de la logique dans les fichiers de vue pour afficher les données d’un modèle complexe, utilisez un [composant de vue](https://docs.microsoft.com/fr-fr/aspnet/core/mvc/views/view-components?view=aspnetcore-6.0), ViewModel ou un modèle de vue pour simplifier la vue.

#### Responsabilité du contrôleur

Les contrôleurs sont des composants qui gèrent l’interaction avec l’utilisateur, fonctionnent avec le modèle et, au final, sélectionnent une vue à afficher. Dans une application MVC, la vue affiche uniquement des informations ; le contrôleur gère les entrées et interactions des utilisateurs, et y répond. Dans le modèle MVC, le contrôleur est le point d’entrée initial. Il est responsable de la sélection des types de modèle à utiliser et de la vue à afficher (ce qui explique son nom, car il contrôle la manière dont l’application répond à une requête donnée)

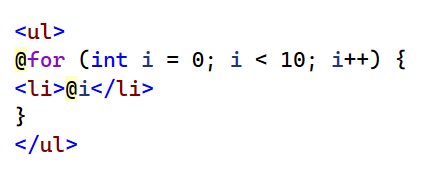
# **Qu'est - ce Razor ?**

Razor est une syntaxe de balisage qui vous permet d’intégrer le code basé sur le serveur (Visual Basic and C#) dans des pages Web.

Le code basé sur le serveur peut créer du contenu web dynamique à la volée, alors qu'une page Web est écrite dans le navigateur. Lorsqu'une page Web est appelée, le serveur exécute le code basé sur le serveur dans la page avant de retourner la page du navigateur. En exécutant sur le serveur, le code peut effectuer des tâches complexes, comme les bases de données accès.

Razor est basé sur ASP.NET, et conçu pour créer des applications web. Il a le pouvoir de balisage ASP.NET traditionnel, mais il est plus facile à utiliser et plus facile à apprendre.

## Razor Syntaxe

Razor utilise une syntaxe très similaire à PHP.

Principales Razor Règles de syntaxe pour C #

● Razor blocs de code sont enfermés dans @ {...}

● Expressions inline (variables and functions) commencer par @

● Les instructions de code se terminent par virgule

● Les variables sont déclarées avec le mot-clé var

● Les chaînes sont placées entre guillemets

● Code C # est sensible à la casse

● fichiers C # ont l'extension .cshtml

Exemple C #

# 

# ASP.NET Razor - C # Variables

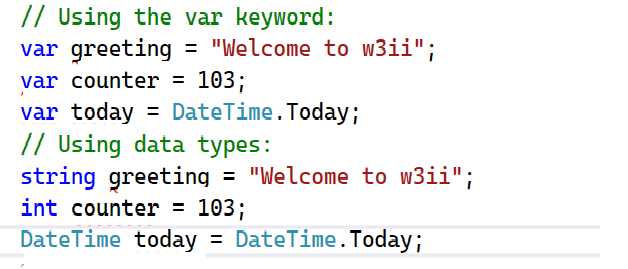
Les variables sont utilisées pour stocker des données.

Le nom d'une variable doit commencer par un caractère alphabétique et ne peut pas contenir des caractères blancs ou réservés.

Une variable peut être d'un type spécifique, ce qui indique le type de données qu'il stocke. Les variables de chaîne des valeurs de chaîne de magasin ("Welcome to w3ii"), variables entières valeurs numériques de magasin (103), les variables de date de valeurs de date de magasin, etc.

Les variables sont déclarées en utilisant le mot - clé var, ou en utilisant le type (if you want to declare the type), mais ASP.NET peuvent généralement déterminer automatiquement les types de données.

Exemples



#### Types de données

Voici une liste des types de données communes :

|  |  |  |
| --- | --- | --- |
| **Type** | **La description** | **Exemples** |
| int | Entier (whole numbers) | 103, 12, 5168 |
| float | nombre à virgule flottante | 3.14, 3.4E38 |
| decimal | Nombre décimal (higher precision) | 1037.196543 |
| bool | Boolean | vrai faux |
| string | Chaîne | "Bonjour w3ii", "John" |

#### Les opérateurs

Un opérateur indique ASP.NET quel type de commande à exécuter dans une expression.

Le langage C # prend en charge de nombreux opérateurs. Voici une liste des opérateurs communs :

|  |  |  |
| --- | --- | --- |
| **Opérateur** | **La description** | **Exemple** |
| = | Affecte une valeur à une variable. | i=6 |
| +  -  \*  / | Ajoute une valeur ou variable.  Soustrait une valeur ou variable.  Multiplie une valeur ou variable.  Divise une valeur ou variable. | i=5+5  i=5-5  i=5\*5  i=5/5 |
| +=  -= | Incrémente une variable.  Décrémente une variable. | i += 1  i -= 1 |
| == | Égalité. Renvoie true si les valeurs sont égales. | if (i==10) |
| != | Inégalité. Renvoie true si les valeurs ne sont pas égales. | if (i!=10) |
| <  >  <=  >= | Moins que.  Plus grand que.  Inférieur ou égal.  Meilleur que ou égal. | if (i<10)  if (i>10)  if (i<=10)  if (i>=10) |
| + | Ajout des chaînes (concatenation) . | "w3" + "schools" |
| . | Point. objets et méthodes distinctes. | DateTime.Hour |
| () | Parenthèse. Les valeurs des groupes. | (i+5) |
| () | Parenthèse. paramètres Laissez-passer. | x= Add(i,5) |
| [] | Supports. Accède valeurs dans des tableaux ou des collections. | name[3] |
| ! | Ne pas. Renverse vrai ou faux. | if (!ready) |
| &&  || | ET logique.  OU logique. | if (ready && clear)  if (ready || clear) |

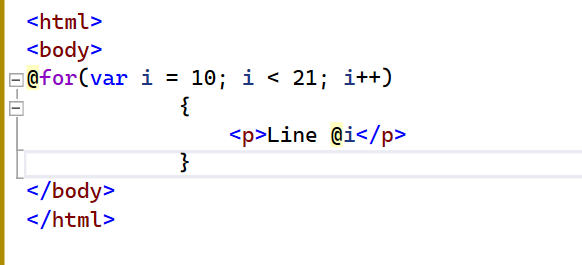
# ASP.NET Razor - C # boucles et tableaux

#### · Boucles :

Si vous devez exécuter les mêmes instructions à plusieurs reprises, vous pouvez programmer une boucle.

Si vous savez combien de fois vous voulez boucle, vous pouvez utiliser une **boucle.** Ce type de boucle est particulièrement utile pour le comptage ou le compte à rebours:

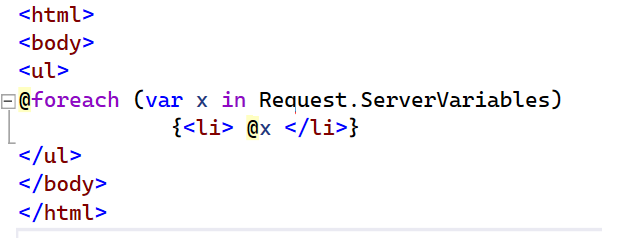
Exemple



Pour chaque boucle, si vous travaillez avec une collection ou un tableau, vous utilisez souvent pour chaque boucle.

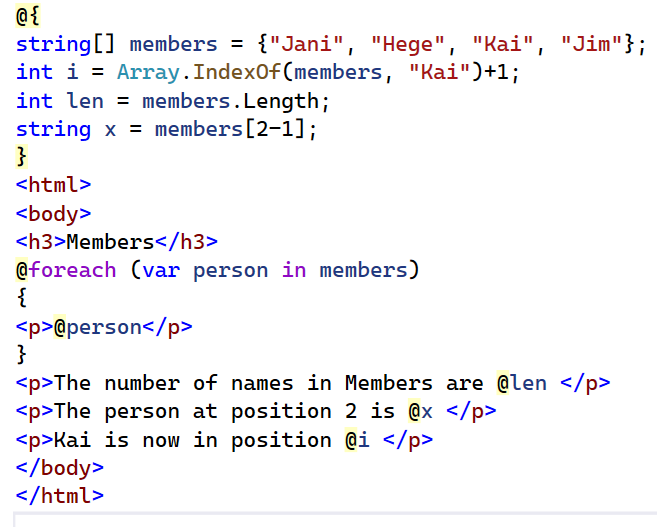
Une collection est un groupe d'objets similaires, et pour chaque boucle vous permet d'effectuer une tâche sur chaque élément. La marche pour chaque boucle dans une collection jusqu'à ce qu'elle soit terminée.

L'exemple ci-dessous des promenades à travers la collection Request.ServerVariables ASP.NET.

Exemple

#### Tableaux :

Un tableau est utile lorsque vous souhaitez stocker des variables similaires, mais ne veulent pas créer une variable distincte pour chacun d’eux :

Exemple

#### C # Conditions Logique

C # vous permet d'exécuter du code en fonction des conditions.

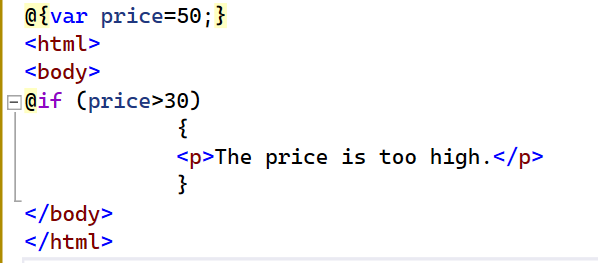
Pour tester une condition que vous utilisez une **instruction if.** L'instruction if renvoie true ou false, en fonction de votre test:

● L'instruction if commence un bloc de code

● La condition est écrit à l'intérieur entre parenthèses

● Le code à l'intérieur des accolades est exécuté si le test est vrai

Exemple



# ASP.NET Core Tag Helpers in Forms:

Ils sont utilisés pour augmenter la productivité des formulaires HTML en créant des formulaires robustes. Ces formulaires sont hautement maintenables même lorsque le logique métier change. Ces assistants de balise génèrent une valeur d'attribut d'action pour les formulaires ainsi qu'un jeton de vérification de demande masqué pour empêcher la falsification de demande intersites.

|  |  |
| --- | --- |
| **Method** | **Description** |
| asp-controller | Spécifiez le contrôleur à cibler en fonction du routage de l'application. S'il est omis, le contrôleur actuel est utilisé. |
| asp-action | Spécifiez la méthode d'action à cibler en fonction des itinéraires dans l'application. S'il est omis, l'action en cours est utilisée. |
| asp-route-\* | Spécifiez la valeur de segment supplémentaire pour l'URL. Par exemple , asp-route-id est utilisé pour fournir une valeur au segment 'id'. |
| asp-route | Spécifiez le nom de la route à cibler pour générer l'attribut d'action. |
| asp-area | Spécifiez le nom de la zone à cibler. |
| asp-antiforgery | Génère un jeton de vérification de demande caché pour empêcher la falsification de demande intersites. Il est utilisé avec l'attribut  [ValidateAntiForgeryToken] dans la méthode d'action HTTP Post. |

Exemple :

#### ASP-Antiforgery Attribute

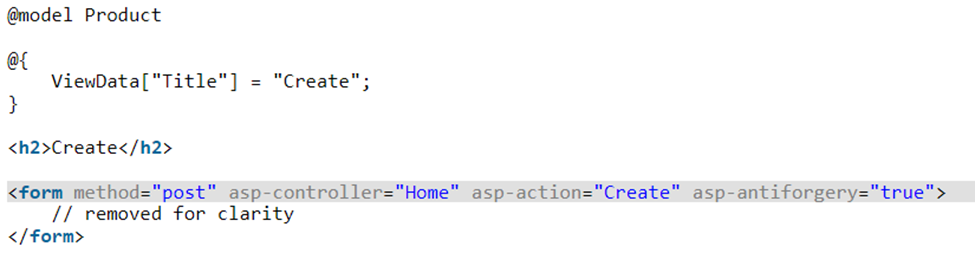
L’attribut asp-antiforgery génère un jeton de vérification de demande masqué appelé antiforgerytoken. Ce jeton empêche les attaques **de falsification de requête intersite (CSRF)**. Lorsque vous utilisez cet attribut dans votre élément Form, ASP.NET Core fait deux choses :

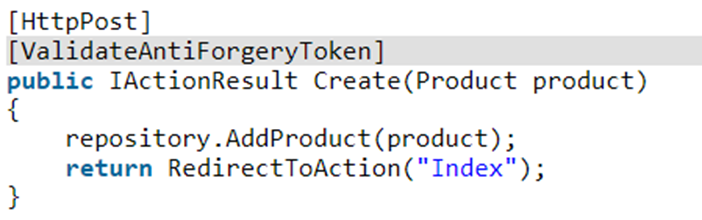
* 1. Ajoute un jeton de sécurité dans un élément d'entrée masqué au formulaire.
* 2. Ajoute un cookie à la réponse.

L'application traitera la demande uniquement si elle contient à la fois le cookie et la valeur cachée du formulaire, à laquelle le site malveillant ne peut pas accéder, et donc CSRF est empêché.

Pour utiliser cette fonctionnalité, ajoutez l'assistant de balise - asp-antiforgery="true"à l'élément de formulaire et ajoutez l'attribut [ValidateAntiForgeryToken] sur la méthode Action.

Accédez à la vue Créer et ajoutez l' **attribut asp-antiforgery** au formulaire :



Ajoutez ensuite l' attribut [ValidateAntiForgeryToken] sur la version HttpPost de la méthode Create Action . J'ai mis en surbrillance cette ligne de code (voir le code ci-dessous).

#### ASP.NET Core Label Tag Helper

Il définit l'attribut for pour une étiquette HTML. Nous ajoutons l'attribut "asp-for" sur l'étiquette qui crée l'attribut for de l'étiquette. Pour tester cela, modifiez les étiquettes données sur la vue Créer pour inclure l'attribut "asp-for" comme indiqué ci-dessous :

#### ASP.NET Core Input Tag Helper

Lorsque nous ajoutons l' attribut **asp-for** à l'entrée, il définit les attributs name, Id, type et value de l'élément Input à l'expression du modèle.

Construisons un exemple : dans la vue Créer , utilisez l' attribut asp-for pour les éléments d'entrée, comme indiqué dans le code en surbrillance ci-dessous.

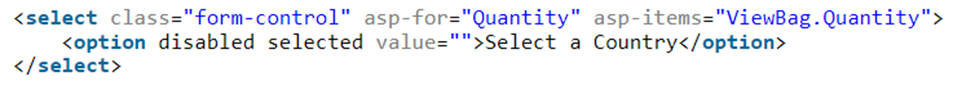


#### ASP.NET Core Select Tag Helper

Là, **Select Tag Helper** spécifie le nom de la propriété du modèle et les éléments d'option pour l'élément html select (liste déroulante). Il y a 2 attributs qui peuvent être appliqués à un élément select qui sont donnés dans le tableau ci-dessous.

|  |  |
| --- | --- |
| **Nom** | **La description** |
| asp-for | Définit l'attribut id et name de l'élément Select sur le nom de la propriété du modèle. |
| asp-items | Spécifie la source des valeurs pour les éléments d'option contenus dans l'élément Select. |

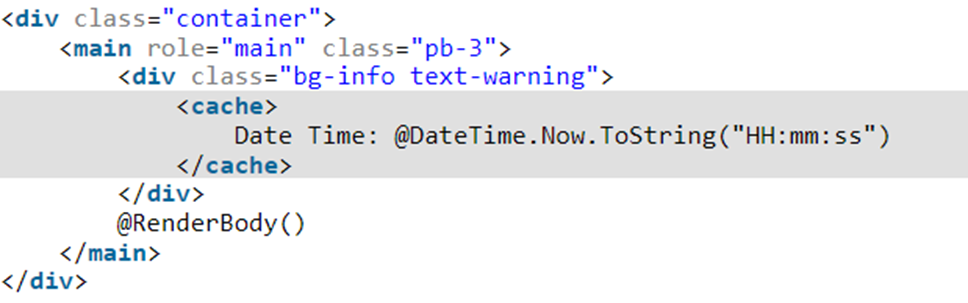
Modifiez le code de l'élément de sélection de la vue Créer qui utilise désormais l’attribut **asp-items :**

****

## Cache Tag Helper

ASP.NET Core Cache Tag Helper est utilisé pour mettre en cache le contenu du fournisseur de cache afin d'améliorer les performances de l'application.

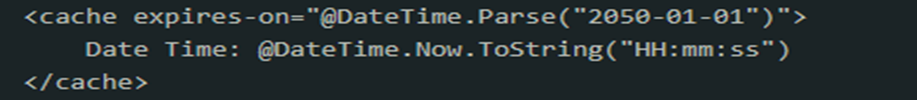
Accédez au fichier \_Layout.cshtml et ajoutez l’élément de cache pour mettre en cache l'heure actuelle en heures-minutes-secondes comme indiqué ci-dessous :



Certains attributs importants pour **Cache Tag Helper** sont donnés ci-dessous :

|  |  |
| --- | --- |
| **Name** | **Description** |
| **expires-on** | Spécifiez une heure absolue dans la valeur datetime à laquelle le cache expire. |
| **expires-after** | Spécifie l'heure relative dans la valeur TimeSpan à laquelle le cache expire. |
| **expires-sliding** | Le temps de glissement est la période écoulée depuis la dernière utilisation. |
| **vary-by-query** | Spécifie la clé de la chaîne de requête qui sera utilisée pour gérer  les différentes versions du cache |
| **vary-by-cookie** | Spécifie le nom du cookie qui sera utilisé pour gérer  les différentes versions du cache. |
| **vary-by** | Spécifie une clé pour gérer différentes versions du cache. |

Modifiez maintenant le code de cache dans la vue \_Layout.cshtml pour utiliser l' attribut expires-on :

****

# liste déroulante dynamique

**Qu'est-ce que ViewBag et ViewData dans Asp.Net Core ?**

ViewBag et ViewData sont tous deux properties disponibles dans l'espace de Microsoft.AspNetCore.Mvc noms. ViewBag renvoie *dynamic* le type de données tandis que ViewData renvoie le *ViewDataDictionary* type.

Pourquoi utilisons-nous ViewBag et ViewData dans Asp.Net Core ?

ViewBag et ViewData sont tous deux utilisés pour transmettre des données de la méthode d'action du contrôleur à sa vue correspondante ou à la page de disposition associée à la vue.

ViewBag et ViewData sont utilisés pour :

· Transmettre les données de la méthode d'action à la vue associée

· Transmettre les données de la méthode d'action à la mise en page associée

· Transmettre les données de la vue à son fichier de mise en page

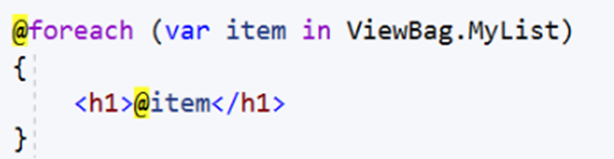
· vue partielle sur la vue

**Comment obtenir de la valeur à partir de ViewBag**

Obtenir la valeur de ViewBag est très simple sur une page cshtml. Nous pouvons obtenir la valeur ViewBag en utilisant la syntaxe du langage Razor.

*@ViewBag.PropertyName*

Si le ViewBag contient une collection de données, l'itération peut être appliquée directement.



**ViewData dans le noyau asp.net**

ViewData est une propriété getter et setter disponible dans l' espace de **Microsoft.AspNetCore.Mvc** noms. ViewData est également disponible dans tous les contrôleurs de l'application Asp.Net Core. ViewData fonctionne avec [ViewDataDictionary](https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.viewfeatures.viewdatadictionary?view=aspnetcore-3.1).Ce ViewDataDictinonary fonctionne sur le concept clé-valeur.

La clé est sous la forme d'un string et la valeur est sous la forme d'un object.

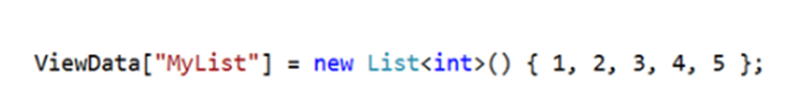
**Comment attribuer une valeur à ViewData :**

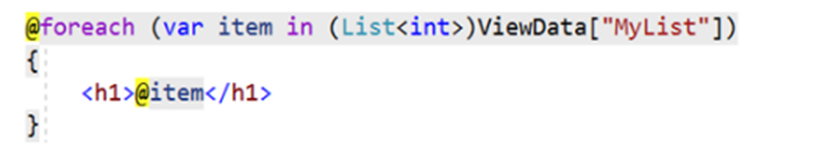
Nous attribuons généralement une valeur à ViewData soit dans un contrôleur, soit dans le fichier de vue. Dans le contrôleur, nous pouvons utiliser ViewData directement mais sur le fichier de vue, nous devons utiliser la syntaxe razor.

*ViewData["PropertyName"] = Valeur ;*

Si nous voulons utiliser la valeur ViewData directement sur la page d'affichage

@ViewData["*PropertyName"]*

Attribuez la liste à ViewData :

Et maintenant itérons sa valeur sur le fichier de vue :

# liste déroulante statique

Le type de données Enum est utilisé pour définir les constantes nommées (c'est-à-dire, la constante ayant un nom et une valeur). Enum peut être utilisé pour remplir un contrôle DropDownList car les membres Enum ont des noms de chaîne et une valeur entière associée à chacun. Voir l'exemple ci-dessous avec toutes les étapes.

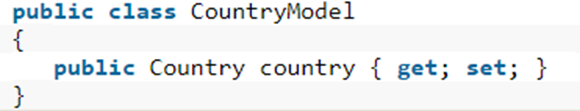
##### Énumération

Créez l'énumération suivante que nous voulons remplir dans la liste déroulante.

****

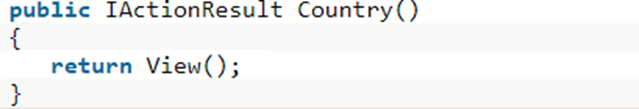
##### Modèle

Dans Model, nous devons définir une propriété de Enum.

****

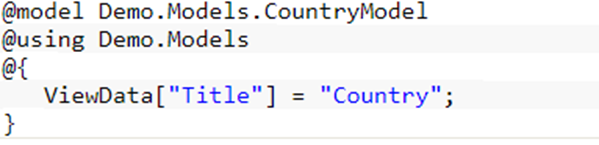
##### Action

Créez un résultat d'action pour afficher la vue.



##### Vue

Dans la vue, appelez le modèle et incluez l'espace de noms dans lequel le County Enum est stocké.



**Choisissez le pays**



# Les vues partielles

Les vues partielles sont une approche différente des Layouts car elles ne sont pas un modèle dans lequel le code sera inclus et parce que son objectif n’est pas que tout soit vu de manière homogène, l’objectif d’une vue partielle est de montrer des données et d’avoir une fonctionnalité, que nous pouvons l'appliquer dans plusieurs parties de la page et que nous ne devrions l'écrire qu'une seule fois.

Par exemple, une application possible de ceci peut être dans un site de vente aux enchères où nous devons placer une section avec les dernières enchères qui ont été appliquées à l'article, une petite photo et le prix, au lieu d'avoir à écrire ce code dans chaque vue, faire l'application d'une vue partielle.

##### Générer une vue partielle

Pour générer une vue partielle, nous devons d’abord décider de ce qu’elle va prendre, puis nous construisons notre logique et la traduisons dans un fichier, ce fichier doit être placé dans le répertoire des vues partagées, c’est-à-dire qu’il est placé dans le dossier partagé de notre application.

Ensuite, si nous voulons utiliser cette vue partielle, nous devons utiliser Html.Partial () de cette manière, nous pouvons générer ce que nous construisons déjà dans le cadre d’une autre vue.

