

ASP.NET MVC: LAYOUT ET VUES PARTIELLES

Sommaire

Introduction.....	1
Le moteur de vue : Razor.....	1
Ajouter du code C# « inline »	1
Ajouter un bloc de code C#	1
Les « Helpers HTML » ou comment économiser son code	2
Créer un Helper HTML	3
La page de disposition : _Layout.cshtml	4
Initialiser le Layout automatiquement : _ViewStart.cshtml	4
Les vues partielles.....	5
Pour résumer	5

Introduction

Lors du développement d'un projet ASP.NET MVC, chaque action de chaque contrôleur peut-être associé à une vue. Jusqu'à présent, vous avez réalisé des vues contenant une structure HTML complète.

Lorsque nous souhaitons réaliser une interface utilisateur, on se rend vite compte que réécrire les parties HTML communes peut devenir un travail fastidieux et difficilement maintenable tant le projet prend de l'ampleur.

ASP.NET MVC dispose d'un mécanisme permettant de répondre à cette problématique. Ce mécanisme va impliquer plusieurs nouveaux fichiers que nous allons découvrir dans ce document.

Le moteur de vue : Razor

Un moteur de vue est une bibliothèque offrant une syntaxe simplifiée pour accéder aux objets générés par le code back-end. Il permet notamment de mixer du code « client » (html, css...) avec du code « serveur » (c#, php, java...).

Le moteur de Vue par défaut d'ASP.NET MVC est **Razor** :

<https://docs.microsoft.com/fr-fr/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-c>

Avec Razor, il est possible de mixer le code HTML des vues avec du code C#.

Le code C# sera alors encapsulé dans des blocs de code qui seront initialisés avec le caractère « @ ».

Ajouter du code C# « inline »

L'exemple suivant démontre comment insérer un appel simple à du code C# à l'intérieur d'une vue :

```
<p>Je suis une vue. Aujourd'hui nous sommes le @DateTime.Now.ToLongDateString()</p>
```

Le code C# est représenté en bleu. Il est possible d'appeler n'importe quelle classe/méthode du Framework selon les besoins. Remarquez qu'il n'y a pas de « ; » à la fin de l'instruction C#.

Ajouter un bloc de code C#

L'exemple suivant illustre comment insérer un bloc de code C# à l'intérieur d'une vue :

```
<div>
<p>Je suis une vue.</p>
@{
    int number = 10;
    string dateDuJour = DateTime.Now.ToLongDateString();
}
<p>Aujourd'hui, nous sommes le @dateDuJour et il y a @number pommes dans le panier.</p>
</div>
```

Le code C# est ici encapsulé dans un bloc commençant par « @{ » et se terminant par « } ».

Notez cette fois-ci, la présence du « ; » à la fin de chaque instruction C# se trouvant dans le bloc.

Remarquez également la réutilisation des variables déclarées via l'utilisation « inline » de C#.

Les « Helpers HTML » ou comment économiser son code

ASP.NET MVC propose un mécanisme de génération de code HTML : les « Helpers ». Ils permettent d'écrire moins de code tout en gardant un fort contrôle sur le HTML.

Par exemple, au lieu d'écrire :

```
<input type="text" id="prenom" value="prenom" />
```

Nous pouvons utiliser le Helper TextBox de la manière suivante :

```
@Html.TextBox("prenom", "Mickael")
```

Ce qui va générer le code HTML suivant:

```
<input type="text" id="prenom" name="prenom" value="Mickael" />
```

Remarquez que le Helper a automatiquement renseigné l'attribut "id" et l'attribut "name" avec le paramètre que nous lui avons passé, ce qui est très pratique !

Pour tous les Helpers :

- Le 1^{er} paramètre (en bleu) correspond au nom du champ (l'attribut « id » et « name » du contrôle HTML généré)
- Le 2nd paramètre (en vert) reçoit généralement la valeur du champ (l'attribut « value » du contrôle HTML généré).
- Il existe généralement plusieurs surcharges pour chaque Helper, n'hésitez pas à les parcourir.

Ci-dessous un tableau présentant les principaux Helpers disponibles avec le moteur de vue Razor :

Helper	Description
Html.ActionLink	Génère un lien vers une méthode (action) de contrôleur avec la balise <a href>
Html.BeginForm	Génère une balise de formulaire <form>
Html.CheckBox	Génère une case à cocher <input> de type "checkbox"
Html.DropDownList	Génère une liste déroulante de type <select>, <option>
Html.Hidden	Génère un champ caché <input> de type "hidden"
Html.ListBox	Génère une liste déroulante multi-sélectionnable de type <select>, <option>
Html.Password	Génère une zone de texte permettant de saisir un mot de passe <input> de type "password"
Html.RadioButton	Génère un bouton radio avec une balise <input> de type "radio"
Html.RouteLink	Fait la même chose que Html.ActionLink à part qu'il utilise une route en entrée
Html.TextArea	Génère une zone de texte modifiable sur plusieurs lignes avec la balise <textarea>
Html.TextBox	Génère une zone de texte modifiable sur une seule ligne avec la balise <input> de type "text"

Il existe d'autres Helpers que les Helpers HTML : Les Helpers URL, qui font la même chose que le Helper ActionLink à ceci près qu'ils ne génèrent pas la balise <a> mais seulement le lien correct en utilisant le mécanisme de Routing d'ASP.NET MVC.

Helper	Description
Url.Action	Génère une URL en utilisant le mécanisme de routing d'ASP.NET MVC
Url.Content	Génère une URL absolue à partir d'une URL relative
Url.RouteUrl	Fait la même chose que Url.Action à part qu'il utilise une route en entrée

Créer un Helper HTML

En plus des Helpers disponibles dans le moteur de vue Razor, il est possible de créer ses propres Helpers réutilisables, ce qui permet d'économiser pas mal de code HTML s'ils sont bien réutilisés.

Voici la procédure à suivre pour créer un fichier destiné à accueillir un ou plusieurs Helpers :

1. A la racine du projet, créer un répertoire « App_Code »
2. Dans ce répertoire, créer un fichier « MyHelpers.cshtml »
3. Vider le contenu du fichier

Une fois le fichier créé, nous allons pouvoir coder notre 1^{er} Helper. Un Helper se construit un peu comme une méthode de classe. Voici la syntaxe : **@helper NomDuHelPer (paramètres) { « code html » }**

Exemple :

```
@helper ShowMessage(string message) {  
    <div class="alert alert-info">  
        <p>  
            <strong>Message : </strong> @message  
        </p>  
    </div>  
}
```

Comme pour une méthode de classe, il est possible de définir 0, 1 ou plusieurs paramètres. Ces paramètres peuvent être de n'importe quel type (string, int, List, un type d'objet, etc...)

Une fois notre Helper créé, il sera directement accessible dans n'importe quelle vue du projet. Il suffira alors de l'appeler en utilisant la syntaxe suivante : **NomDuFichier.NomDuHelPer(paramètres)**

Exemple :

```
<!DOCTYPE html>  
<head>  
    <title>Test Helpers Page</title>  
</head>  
<body>  
    <p>Bonjour</p>  
  
    <!-- Le Helper ShowMessage se trouve dans le fichier "~/App_Code/MyHelpers.cshtml" -->  
    @MyHelpers.ShowMessage("Ceci est un message")  
  
    <p>Au revoir</p>  
</body>  
</html>
```

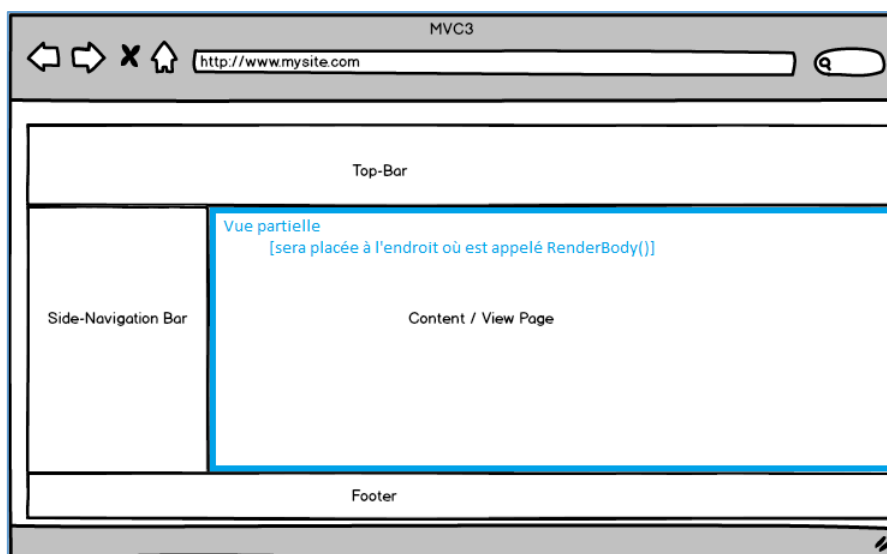
Le code HTML du Helper sera placé exactement à l'endroit duquel il a été appelé.

La page de disposition : `_Layout.cshtml`

Pour répondre à la problématique de l'interface utilisateur évoquée plus haut, ASP.NET MVC permet de créer une **page de disposition** autrement appelée « **Layout** ».

Par convention, le « Layout » principal est défini dans le fichier « `~/Views/_Layout.cshtml` ».

La page de disposition contiendra la structure (le code HTML global) des pages web du projet et encapsulera les vues partielles liées aux contrôleurs. L'appel des vues liées aux contrôleurs se fait au travers d'une méthode « `RenderBody()` » qui sera appelée depuis l'intérieur du Layout. À la compilation, la méthode `RenderBody()` affichera le code HTML de la vue renvoyée par le contrôleur.



Dans l'image ci-dessus, Les éléments « Top-Bar », « Side-Navigation-Bar » et « Footer » font partie du Layout. Le code HTML associé à ces éléments se trouve dans le fichier « `~/Views/_Layout.cshtml` ».

La partie encadrée en bleu, soit l'élément « Content/View Page », Correspond à l'emplacement de la méthode « `RenderBody()` » dans la page de disposition (`_Layout.cshtml`) et chargera à cet endroit la vue renvoyée par le contrôleur.

Initialiser le Layout automatiquement : `_ViewStart.cshtml`

Si ASP.NET MVC identifie un fichier nommé « `_ViewStart.cshtml` » dans le répertoire « `~/Views/` » il sera automatiquement chargé.

Le fichier `_ViewStart` va permettre de définir le Layout par défaut qui sera utilisé par les vues chargées par les contrôleurs.

Par exemple, pour associer la Layout « `~/Views/_Layout.cshtml` », le fichier `_ViewStart` contiendra le code suivant :

```
@{
    Layout = "~/Views/_Layout.cshtml";
}
```

Note : Si aucun répertoire n'est précisé dans le chemin, ASP.NET MVC recherchera le fichier correspondant dans le répertoire « `~/Views/` ».

Dans l'exemple ci-dessus, ASP.NET MVC chargera le fichier « `~/Views/_Layout.cshtml` » s'il existe.

Les vues partielles

Une vue partielle est, comme son nom l'indique, une vue qui ne contient qu'une partie du code HTML d'une page Web. Dans une vue partielle, pas de balise `<html></html>` ni de `<head></head>` et encore moins de `<body></body>`.

Les vues partielles sont destinées à être directement encapsulées dans une autre vue voire dans l'interface utilisateur qui sera implémentée dans la page de disposition du projet (le Layout pour ceux au fond qui n'ont pas suivi).

Pour créer une vue partielle, cocher la case « Vue partielle » dans la boîte « Ajouter → Vue ». Le code HTML généré sera alors exempt de toute structure.

Remarquez l'éventuelle présence d'un bloc C# au début de chaque vue partielle générée.





La variable « Layout » permet de définir la de disposition qui sera utilisée pour la vue courante. Par défaut, la valeur de la variable « Layout » est « null ». Pour associer la vue partielle à une page de disposition, il suffit d'entrer le chemin vers la page de disposition souhaitée (ex « ~/Views/_Layout.cshtml »). Pour Utiliser la page de disposition définie dans le fichier _ViewStart.cshtml, il suffit de commenter ou supprimer la variable « Layout » de la vue partielle.

Il est possible de réutiliser une vue partielle à plusieurs endroits.

Pour résumer

- Le moteur de vue d'ASP.NET MVC s'appelle « Razor »
- Dans une vue, il est possible de mixer code HTML et code C#
- Les Helpers HTML offre une syntaxe simplifiée pour générer des éléments HTML
- Il est possible de créer ses propres Helpers qui seront alors placés dans le répertoire « ~/App_Code/ »
- Le fichier « ~/Views/_Layout.cshtml » contient la structure HTML du site.
- Le fichier « ~/Views/_Layout.cshtml » contient un appel à `RenderBody()`
- `RenderBody()` chargera à son emplacement la vue partielle renvoyée par le contrôleur
- Le fichier « ~/Views/_ViewStart.cshtml » permet de définir un Layout pour tout le site.
- Dans les vues partielles :
 - o La variable « Layout » permet de définir une page de disposition pour la vue courante
 - o Supprimer la variable Layout chargera la page de disposition définie dans _ViewStart.

--- FIN DU DOCUMENT ---

Légende des icônes	
	Information complémentaire
	Point d'attention particulier
	Intervention du formateur possible
	Lien vers une ressource externe

<http://www.arfp.asso.fr>